



CTPU-16-36 CSIC-16-116 TTK-16-47

CheckMATE 2: From the model to the limit

Daniel Dercks^{a,b,1}, Nishita Desai^{c,2}, Jong Soo Kim^{d,e,3}, Krzysztof Rolbiecki^{f,4}, Jamie Tattersall^{g,5},
Torsten Weber^{g,6}

^a*II. Institute for Theoretical Physics, University of Hamburg, Luruper Chaussee 149, D-22761 Hamburg, Germany*

^b*Bethe Center for Theoretical Physics & Physikalisches Institut der Universität Bonn, Nussallee 12, D-53115 Bonn, Germany*

^c*Laboratoire Charles Coulomb (L2C) UMR 5221 & Laboratoire Univers et Particules de Montpellier (LUPM) UMR 5299, CNRS-Université de Montpellier, 34090 Montpellier, France*

^d*Center for Theoretical Physics of the Universe, Institute for Basic Science (IBS), Daejeon, 34051, Korea*

^e*Universidad Autónoma de Madrid, Instituto de Física Teórica, Calle Nicolás Cabrera 13-15, Cantoblanco, 28049 Madrid, Spain*

^f*Institute of Theoretical Physics, University of Warsaw, Pasteura 5, 02-093 Warsaw, Poland*

^g*Institute for Theoretical Particle Physics and Cosmology, RWTH Aachen University, D-52056 Aachen, Germany*

Abstract

We present the latest developments to the **CheckMATE** program that allows models of new physics to be easily tested against the recent LHC data. To achieve this goal, the core of **CheckMATE** now contains over 60 LHC analyses of which 12 are from the 13 TeV run. The main new feature is that **CheckMATE 2** now integrates the Monte Carlo event generation via **MadGraph5_aMC@NLO** and **Pythia8**. This allows users to go directly from a **SLHA** file or **UFO** model to the result of whether a model is allowed or not. In addition, the integration of the event generation leads to a significant increase in the speed of the program. Many other improvements have also been made, including the possibility to now combine signal regions to give a total likelihood for a model.

Keywords: Analysis, Confidence Limits, Monte Carlo, Detector Simulation, Delphes, ROOT, LHC, Recasting, Beyond the Standard Model 12.60.-i

¹daniel.dercks@desy.de

²nishita.desai@umontpellier.fr

³jongsoo.kim@tu-dortmund.de

⁴krzysztof.rolbiecki@fuw.edu.pl

⁵tattersall@physik.rwth-aachen.de

⁶torsten.weber@rwth-aachen.de

PROGRAM SUMMARY

Program Title: CheckMATE

Journal Reference:

Catalogue identifier:

Licensing provisions: none

Programming language: C++, Python

Computer: PC, Mac

Operating system: Linux, Mac OS

Keywords: Analysis, Confidence Limits, Monte Carlo, Detector Simulation, LHC, Recasting, Beyond the Standard Model

Classification: 11.9

External routines/libraries: ROOT, Python, HepMC (optional)

Subprograms used: Delphes

Nature of problem: The LHC experiments have performed a huge number of searches for new physics in the past few years. However the results can only be given for a few benchmark models out of the huge number that exist in the literature.

Solution method: CheckMATE is a program that automatically calculates limits for new physics models. The original version required the user to generate Monte Carlo events themselves before CheckMATE could be run but the new version now integrates this step. The simplest output of CheckMATE is whether the model is ruled out at 95% CLs or not. However, more complicated statistical metrics are also available, including the combination of many signal regions.

Restrictions: Only a subset of available experimental results have been implemented.

Running time: The running time scales about linearly with the number of input events provided by the user. The event generation, detector simulation and analysis of 10000 events needs about 245 s for a single core calculation on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60 GHz with 32 GB RAM.

Important Note

- **CheckMATE** is built upon the tools and hard work of many people. If **CheckMATE** is used in your publication it is extremely important that all of the following citations are included,
 - Delphes 3 [1].
 - FastJet [2, 3].
 - Anti- k_t jet algorithm [4].
 - CLs prescription [5].
 - All experimental analyses that were used to set limits in the study and if the analysis was implemented by non-CheckMATE authors, the relevant implementation reference.
 - MadGraph5_aMC@NLO [6] if MadGraph5_aMC@NLO is used to calculate the hard matrix element events from within CheckMATE.
 - Pythia 8.2 [7] if showering or matching done from within CheckMATE.
 - The Monte Carlo event generator that was used if .hepmc or .lhe files were generated externally.
 - In analyses that use the m_{T2} kinematical discriminant [8, 9] we use the mt2_bisect library [10]. We also include the $M_{T2}^{b\ell}$ and M_{T2}^W derivatives [11].
 - In analyses that use the M_{CT} family of kinematical discriminants we use the MctLib library that includes the following variables, M_{CT} [12], M_{CT} corrected [13], M_{CT} parallel and perpendicular [14].
 - In analyses that use topness variable we use the topness library [15].
 - In analyses that use Super-Razor [16].

Contents

1	Introduction	4
2	General Program Flow	5
2.1	The Python Scaffolding	5
2.2	FRITZ	5
2.3	Event Use or Generation:	7
2.4	Delphes	8
2.5	AnalysisHandler	8
2.6	Analyses	9
2.7	Evaluation and Output	9
3	Full List of CheckMATE Parameters	9
4	Example: Running CheckMATE and Understanding the Results	13
4.1	Benchmark Model	14
4.2	Initialising and Starting CheckMATE	15
4.3	Structure of the Results Folder	19
4.3.1	Folder <code>internal/</code>	19
4.3.2	Folder <code>fritz/</code>	19
4.3.3	Folder <code>mg5amcatnlo/</code>	24
4.3.4	Folder <code>pythia/</code>	25
4.3.5	Folder <code>delphes/</code>	26
4.3.6	Folder <code>analysis/</code>	27
4.3.7	Folder <code>evaluation/</code>	28
5	Available Analyses	31
6	Performance Studies	35
7	Analysis Manager	36
7.1	Prototyping New Analyses	36
7.2	Kinematical Variables	36
8	Summary	36
Appendix A	Installation Instructions	38
A.1	Required Packages	38
A.2	Optional Packages	39
A.3	Installing CheckMATE	40
Appendix B	Statistical Analysis in CheckMATE	40
B.1	1-bin Likelihood and Test Statistics	40
B.2	Confidence Levels and p -values	41
B.3	Model Independent Limits S_{95}	42
B.4	Likelihood	43
Appendix C	Tuning	43
C.1	Electrons	43
C.2	Muons	44
C.3	B -Tagger	44
C.3.1	8 TeV	45
C.3.2	13 TeV	47

1. Introduction

The first years of the Large Hadron Collider (LHC) running have been a triumph with the notable discovery of the Higgs boson being a particular highlight [17, 18]. Whilst no other new states have been found yet, a large number of searches for physics beyond the Standard Model (BSM) have been performed. This dataset has profound implications for many of these BSM theories but the experimental collaborations themselves only have limited resources to investigate the many models on the market. Therefore it is imperative that theorists take up the task of testing their models against the wealth of data available.

To help with such studies, a number of tools have now been made public that allow for easy and fast model testing. One class of programs are based on the so-called ‘simplified models’ [19] approach. Here the tools make use of the actual LHC limits on particular topologies that are commonly seen in BSM. The limits are then adjusted to account for the correct branching ratios in the actual model under test. The advantage of such an approach is that these programs are very quick to return an answer. However the big disadvantage is that if the new physics model contains final state topologies not originally tested, the limit will be severely weakened compared to the true result. Unfortunately this is very common in realistic models that may have longer or asymmetric decay chains. Examples of tools that use simplified models are **SModelS** [20, 21] and **FastLim** [22] for supersymmetry and **XQCUT** [23] for models with extended quark sectors. Furthermore, if limits from a simplified model analysis are used for a different model (e.g. one predicting different angular correlations or extra intermediate particles) the results may not be accurate. This severely limits the applicability to classes of models with particles of exactly the same spin and an identical decay chain to the one originally tested.

The second approach⁷ for theorists to test models against the LHC data is to essentially follow the same procedure that the experimental collaborations perform themselves, or to “recast”.⁸ Here, Monte Carlo (MC) events are simulated for the particular model under test and these events are then passed through a detector simulation that returns reconstructed final state objects. The same experimental cuts used by the experiment are then placed on these objects and limits are placed on the number of events seen in pre-defined signal regions. This chain is validated against benchmark models provided by the experiments. Once validated, the recast analysis can be used to test any model by changing the MC events supplied. Whilst this method has the disadvantage of being slower than the simplified model approach, the big advantage is the generality that allows for a large range of different theories to be tested. Examples of tools that use recast analyses are **CheckMATE** [26, 27] and **MadAnalysis** [28, 29] while the newly released **Contur** [30] uses the **RIVET** [31] library of Standard Model measurements.

So far all the recast based tools require the user to provide externally generated events that have been showered, hadronised and already have any required underlying event modelling. This has severe performance drawbacks aside from the obvious extra effort for the end user to generate these events. Firstly, the Monte Carlo events must be stored to disk and this process involves writing and re-reading several 10’s or even 100’s GB for the required statistics.⁹ Consequently, even when we are just testing a single model point the Monte Carlo events already require significant amounts of free space. However, a greater issue is if we want to test many model points on a large cluster. In a cluster architecture where many jobs run from a single hard drive, the reading and writing of events is already the limiting factor for just 2–3 simultaneous processes.

To solve this problem, we present **CheckMATE 2** which integrates both **MadGraph5_aMC@NLO** and **Pythia 8** into a complete model testing loop. As a consequence, the user only has to provide an **SLHA** file [32, 33] in the case of supersymmetry or a **FeynRules** [34, 35] **UFO** [36] model file for other models and the event generation is then taken care of internally by **CheckMATE**. We also note that the improved efficiency due to skipping the event file storage results in a single process running up to $\sim 40\%$ faster. More importantly

⁷Another approach, that we do not discuss here, is training a machine learning algorithm on already tested models which has been pioneered by **SUSY-AI** [24].

⁸“To Recast” is a verb coined from Ref. [25] and is now increasingly used as shorthand to describe the full process of reproducing an experimental analysis.

⁹We find e.g. that 10,000 hadronised events require between ~ 1 and ~ 10 GB.

however, is the far improved cluster processing. That means running many simultaneous jobs from a single hard disk is now made possible without any reduction in performance.

The incorporation of the event generation is not the only improvement in **CheckMATE 2** though. Over 60 experimental analyses from the LHC collaborations are now available covering the 7, 8 and 13 TeV runs and more are continually being added. In addition, **CheckMATE 2** now contains 14 TeV high luminosity LHC analyses as well. Consequently the user can now go beyond simply setting limits on models with the current data and investigate what is the ultimate LHC reach to the model of interest. Further improvements have also been made to the **AnalysisManager** introduced in Ref. [27] so that users can more easily investigate new ideas for LHC searches. These include more LHC kinematical observables being included and additional tools that make the statistical analysis of the results easier.

We begin this paper in Section 2 by giving an overview of the individual building blocks that make up the **CheckMATE** program. We follow in Section 3 by giving the full list of available **CheckMATE** parameters that may be useful to more advanced users. To more easily explain how **CheckMATE** is used, we then provide an example run in Section 4 that also discusses in more detail some of the most commonly used options. A brief summary of the currently available analyses is provided in Section 5. Section 6 investigates the performance improvements of the new version of **CheckMATE** over the old in more detail while Section 7 explains the improvements to the **AnalysisManager**. Finally, we summarise in Section 8. Appendix A gives installation instructions, Appendix B covers the statistical methods used in **CheckMATE** in more detail and Appendix C describes the updated lepton identification performance and b -tagging efficiencies now used in **CheckMATE**.

2. General Program Flow

CheckMATE incorporates many individual modules which cover the steps necessary for model testing. A flowchart illustrating the modules and the data passed internally between them is given in Figure 1. The modules are embedded in a **Python** scaffolding which handles the user prompts, the file I/O and the setup of the core modules which we describe in more detail below.

2.1. The *Python* Scaffolding

The initialisation is performed by a **CheckMATE Python** script which reads in user input (either via a file or command line), and writes the configuration files required by the **FRITZ** and **AnalysisHandler** modules to set up event generation, showering and detector simulation followed by recasting of the experimental analysis. The computation-heavy parts are taken over by the individual modules coded in **C++** described below, which call as required external libraries or programs (viz. **MadGraph5_aMC@NLO**, **Pythia8**, **HepMC**, **Delphes** and **ROOT**). After the recast analyses are run, the results are again collected and processed by the **Python** script to check against published 95% confidence level (CL) upper limits. The ratio of expected signal to the 95% CL upper limit and whether the point is still allowed is reported as the default primary result. Further options for statistical evaluation are discussed in Section 2.7.

2.2. *FRITZ*

FRITZ¹⁰ (Flexible Rapid Interactive Tool Zipper) denotes the core **C++** program of **CheckMATE**. Depending on the provided data and settings, it connects to and runs **MadGraph5_aMC@NLO**, **Pythia8** and **Delphes**, followed by the **AnalysisHandler** and all the analyses requested by the user. Except for the LHE files produced by **MadGraph5_aMC@NLO**, intermediate data, e.g. the simulated Monte Carlo events generated by **Pythia** and/or the detector level objects produced by **Delphes**, are passed on-the-fly between the individual modules. This is a great improvement on the original **CheckMATE** version 1 in which the generated events as well as the detector level objects had to be stored and then re-read from hard disk. Given that a typical BSM Monte Carlo event file including hadronised final states and a sufficiently high statistical sample easily

¹⁰The name *Fritz* is derived from a German chess program of the same name, see Ref. [37, 38], and the very first chess computer program one of the authors (DD) played.

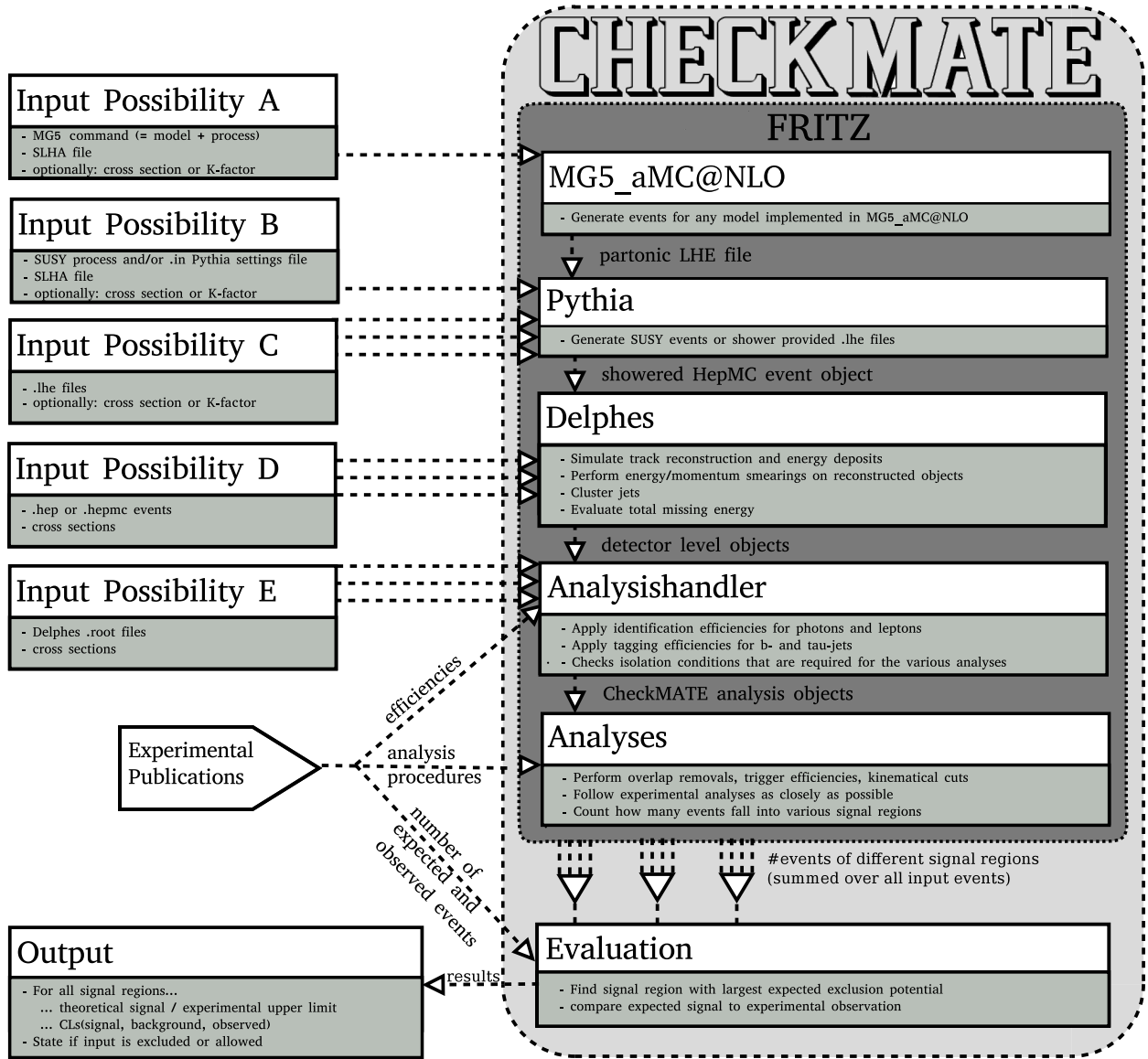


Figure 1: Flow chart to demonstrate the chain of data processing within CheckMATE.

reaches file sizes of several GB, significant time is saved in I/O by this improvement¹¹ besides removing the requirement for large storage. However, if the user requires so, the intermediate objects can be written to the disk using the switches `WritePythiaEvents` and `WriteDelphesEvents`, cf. Section 3.

2.3. Event Use or Generation:

One of the core parts of Monte Carlo based collider phenomenology is the simulation of final state configurations that would be produced in a collider experiment if a particular model of BSM physics was true. In the first version of `CheckMATE`, the event generation had to be done externally by the user. MC event files and the corresponding cross section — either from the same event generator or from an external cross section calculator like `Prospino` [39–43] or `NLLFast` [39, 40, 44–48] — were mandatory input parameters which were then processed via `Delphes` within `CheckMATE`. Besides the practical inconvenience that every `CheckMATE` user had to use an external event generator, the forced split between event generation and detector simulation/analysis also yields a computational disadvantage as already explained above. Consequently, the new `CheckMATE` version now provides an automatic link to both `MadGraph5_aMC@NLO` [6] and the `Pythia8` [7] event generation. With this new functionality, `CheckMATE` provides different types of modes to either run `MadGraph5_aMC@NLO` and `Pythia8` or simply use already generated event files:

Provide an externally produced .hepmc or .hep event file: We first emphasise that if the user wishes to provide Monte Carlo events in either `.hepmc` or `.hep` format to `CheckMATE`, this option is still supported. `CheckMATE` will then pass these events directly to `Delphes` for detector simulation.

Generate events entirely using Pythia8: `Pythia8` is capable of generating events for BSM models followed by parton showering and hadronisation of the final state. This functionality can be accessed by `CheckMATE` in two different ways.

The first possibility is to provide the `Pythia8` setup via an `.in` file which uses the `Pythia8` internal syntax, see refs. [7, 49, 50], to set the internal parameters. This mode allows for the full flexibility of the `Pythia8` program as all parameters can be changed via this input file method. Most importantly, the `.in` file is used to define the model and the list of processes which should be generated. All model parameters (e.g. couplings, masses, widths, branching ratios etc.) must be provided in the input file. If a supersymmetric (SUSY) process is desired, the `SLHA` file [32, 33] which specifies the parameter point must be provided within the input file in the standard `Pythia8` syntax.

Given the popularity of SUSY models, we provide an additional shortcut to generate SUSY processes without the need to provide a full input card. All showering and hadronisation parameters will then be taken from the default `Pythia` settings. The process can be set directly using a simplified `MadGraph5_aMC@NLO`-like syntax e.g. `Pythia8Process: p p > go go` to initiate gluino pair production. A further shorthand to refer to classes of SUSY particles is also available. The full list of available processes using this running mode is given in Section 3.

In both cases, `CheckMATE` will use `Pythia8` to simulate the given processes and will directly perform detector simulation followed by applying analysis routines. As explained above the events are not stored unless explicitly demanded by the user.

By default, `CheckMATE` uses the cross section and statistical uncertainty provided by `Pythia8`. A user can also provide a cross section or a K -factor, i.e. $\sigma_{\text{NLO}}/\sigma_{\text{LO}}$, calculated via an external code (see Section 3).

Generate events with MadGraph5_aMC@NLO, shower and hadronise with Pythia8: This option allows a user to both generate the hard matrix element using `MadGraph5_aMC@NLO` by providing a process card (and parameter and run cards if necessary) followed by showering and hadronisation of the resultant LHE files via `Pythia`. This allows users to generate Monte Carlo events for a huge range of BSM models using the UFO file format [36].

¹¹We show later that we can gain a factor of 3 in speed between `CheckMATE` version 1 and `CheckMATE` version 2 depending on the details of the benchmark model and the number of parallel runs. Details can be found in Section 6.

Shower and hadronise externally provided .lhe files: If a user wishes to calculate the matrix element and generate events with a different generator than `MadGraph5_aMC@NLO`, e.g. `WHIZARD` [51, 52] or `CalcHep` [53], this is also possible as long as the program is able to output event files in the `.lhe` format. The advantage of this approach is that the hard process simulation and the parton showering and hadronisation steps can be performed rather independently and it is usually the latter two which take the most computational effort and require most disk space. `CheckMATE` can take lightweight `.lhe` files produced by an external tool and use `Pythia8` to shower/hadronise on-the-fly as described above. A default `.in` file is then used to set the collision energy and to retain default `Pythia8` settings but if necessary users can also provide their own file instead (if e.g. a user wishes to turn off multi-particle interactions etc.)

`Pythia8` internally generates random numbers starting from a default seed. The user can further provide the random number seed in two ways — first by setting an integer seed via `RandomSeed: X` and second by providing a binary state for the `Pythia8` run using the `Pythia8Rndm` key in the `CheckMATE` input file. Each `Pythia8` run writes out the random state at the beginning and end of the run in the files `rndm-init.dat` and `rndm-end.dat` respectively. These can then be used either to reproduce a run or while adding events to ensure there is statistical independence of the two runs.

2.4. Delphes

The events that were either generated internally or provided by the user are then passed to the detector simulation program `Delphes` [1]. In `CheckMATE 2.0`, contrary to the original `CheckMATE` described in Ref. [26], `Delphes` is now only used to simulate the calorimeter and tracking using the standard detector settings for ATLAS and CMS along with jet reconstruction. The identification efficiencies for final state particles are now performed internally by `CheckMATE`. In addition, the output event object is extended to include generator level particles which are required for external b - and τ -tagging.

The results of the detector simulation are passed as ROOT objects and are typically not saved on disk (unless explicitly requested) but instead immediately processed by the analysis framework described below. By setting `WriteDelphesEvents` to `True` (see also Section 3), a `.root` output file can optionally be created and used as an event file in a future `CheckMATE` run, see also Figure 1.

If a user decides to test only ATLAS or only CMS analyses, `Delphes` needs to run only once per event. Otherwise, each input event is processed by two independent `Delphes` runs and independent detector level objects for ATLAS and CMS analyses are respectively created.

2.5. AnalysisHandler

The detector level objects created in the previous step contain reconstructed electrons, muons, photons, jets, tracks, clustered calorimeter cells and the missing transverse momentum vector. These are now further processed by a so-called `AnalysisHandler` before being passed to the actual data analysis codes.

Depending on the list of analyses selected by a user, final state objects are tested against a list of isolation, identification and tagging conditions which set individual tags on those candidates which pass the respective constraints. For that purpose, `CheckMATE` first determines which constraints have to be considered in order to provide all analyses with the required information on the final state objects. As an example, let us assume a user chose three analyses out of which two require tight leptons (usually this means electrons or muons at the LHC) and the third one vetoes events with medium leptons. Then the `AnalysisHandler` using a lepton list passed from `Delphes` will create three new lepton lists, corresponding to different identification working points: `tight` \subset `medium` \subset `loose`. The decision about the assignment for a particular lepton is based on its momentum and takes into account identification efficiencies reported by the experiments, see also Appendix C. Finally, isolation criteria will be checked for each of the leptons. The `AnalysisHandler` will also apply simplified b - and τ -tagging algorithms, if required.¹²

¹²In the first `CheckMATE` version, all these steps have been performed within the `Delphes` framework by generating a run-specific `Delphes` detector card. That made it impossible to re-use `Delphes` output files, either within `CheckMATE` to a *posteriori* test additional analyses or outside `CheckMATE` with a different `Delphes`-based analysis framework like e.g. `MadAnalysis5` [28]. Therefore, the `CheckMATE` and `MadAnalysis5` teams mutually agreed to switch to a final state post-processing outside `Delphes`, see also [54].

CheckMATE uses a set of independent **AnalysisHandlers** with individual tagging efficiencies depending on the list of analyses chosen by the user. Due to the evolution of the reconstruction and identification algorithms, there exist separate **AnalysisHandlers** for analyses performed on 7, 8 and 13 TeV data and for projective studies at 14 TeV centre-of-mass energy. For each of those there is an independent ATLAS and CMS version. The tunings for 7 and 8 TeV are described in more detail in Ref. [26] while the updates for 13 TeV are given in Appendix C.

2.6. Analyses

After all detector level objects have been properly prepared by the **AnalysisHandler(s)**, these are processed event by event by each analysis selected by a user when starting **CheckMATE**. They are internally coded in a framework that allows for an easy extension to new upcoming experimental results and allows users to easily update given analyses or implement their own, see also the original **CheckMATE** publication in Ref. [26]. For a detailed description of the **AnalysisManager** framework see [27].

Each event is processed by checking isolation criteria, removing overlapping objects and implementing the cuts that define the signal regions. The analysis program then determines how many events in total satisfy certain signal region criteria and stores this information in a human-readable output for each separate input event file or alternatively each separate **Pythia** run. In addition we also store the actual number of Monte Carlo events (i.e. the efficiency times acceptance $\mathcal{A} \times \epsilon$) and the number of signal events $S = \mathcal{L}_{\text{int}} \times \sigma \times \mathcal{A} \times \epsilon$, where \mathcal{L}_{int} is the integrated luminosity, and the cross section σ is provided either internally by **Pythia8** or externally by the user. In case of weighted events, the event weights are taken into account while calculating the efficiency.

2.7. Evaluation and Output

The final step of the program consists of a statistical evaluation of the results. For each individual signal region of every chosen analyses, the total number of expected signal events S is determined by summing up the results from each input event file (or event generation run) as explained later in Section 4.3.7. The total 1σ uncertainty ΔS on this number is determined from both the statistical uncertainty, given by the number of Monte Carlo events, and the systematic uncertainty, which is estimated from the total uncertainty on the signal cross section given by the user as an optional parameter. These numbers are compared to the results published in the respective experimental search. There are two possible ways to perform the comparison (see also Ref. [26]): in the standard approach, the number S is tested against a pre-calculated model independent 95 % CL limit S_{95} . Alternatively, a user can choose to calculate the proper CL_S value by folding in the uncertainty on the model prediction. This is not only more accurate but allows for testing against limits other than 95 % CL. In both cases, **CheckMATE** calculates the observed CL_S value using the actual data recorded and the expected CL_S value which assumes the observed data equals the background expectation. In contrast to the earlier **CheckMATE** version, the current version uses its own routines to perform the statistical calculations using the algorithm explained in Appendix B. Besides those two means of model testing, **CheckMATE** optionally allows for the calculation of the likelihood of the final result. This allows for model parameter fits and corresponding confidence limit evaluations. The calculation of the likelihood is explained in detail in Appendix B.

The results of the evaluation are then output for every signal region of every analysis. If multiple signal regions are considered, the model point is determined to be “excluded” or “allowed” based on the signal region which has the best sensitivity assuming background-only hypothesis evaluated by the experiment. This is done to avoid erroneously ruling out a model point due a downward fluctuation of the observed events in a signal region that is not expected to be sensitive in the first place.

3. Full List of CheckMATE Parameters

There exist many optional parameters within **CheckMATE** which can change the standard behaviour of the code and here we describe their usage. These parameters can either be provided via the input file or alternatively, **CheckMATE** can be set up directly within the command line by adding **-parameter value** pairs

after the `./CheckMATE` command. The second alternative is unfortunately only possible for a setup with a single process as only one `-p` command can be provided. If more than one process needs to be analysed in the same run, one either has to use the input file or use the `add` feature described below.

Analyses: The full list of currently implemented analyses is given in Section 5. The following examples show how to specify which of these **CheckMATE** should take into account using an input file (command line),

- **Analyses:** `atlas_1404.2500` (`-a atlas_1404.2500`) tests only the analysis `atlas_1404.2500`.
- **Analyses:** `8TeV` (`-a 8TeV`) tests all implemented analyses which correspond to $\sqrt{s} = 8$ TeV. Alternative values are `7TeV`, `13TeV` and `14TeV`.
- **Analyses:** `atlas&8TeV` (`-a atlas&8TeV`) tests all implemented ATLAS analyses of the given centre-of-mass energy. Similarly for `cms`.

The above specifiers can be combined via a simple separation with commas, e.g. `-a cms&8TeV, atlas_1404.2500` tests all 8 TeV CMS analyses and the single ATLAS analysis `atlas_1404.2500`. All analyses combined this way *must* correspond to the same center-of-mass energy, otherwise **CheckMATE** will abort.

Invisible PIDs: Physics beyond the Standard Model which addresses the dark matter problem often predicts the existence of one or more stable, light particles that only interact weakly with ordinary matter. Whilst **Delphes** automatically identifies all neutral particles in the Minimal Supersymmetric Standard Model (MSSM), other BSM particles have to be explicitly declared as invisible as they are otherwise considered as exotic hadrons which deposit their energy in the hadronic calorimeter. As an example, a Higgs portal model with a stable scalar would require placing in the parameter file the following setting: `Invisible PIDs: 35`.

Result file columns: **CheckMATE** stores the results of its analyses in various files to allow for a detailed investigation how the final result was determined. The standard content of these result files — we describe them in more detail in section 4 — is adaptable such that more intermediate results may be stored. By setting the corresponding `ResultFileColumns` parameter to `a,b,c,...` the corresponding file(s) are set to contain respective information *a,b,c*. `EventResultFileColumns` and `ProcessResultFileColumns` can be taken out of the following set:

`analysis, sr, totalmcevents, totalnormevents, totalsumofweights, totalsumofweights2, signalsumofweights, signalsumofweights2, signalnormevents, signal_err_stat, signal_err_sys, signal_err_tot.`

The names are mostly self-explanatory; the prefix `total-` refers to the full input sample whereas `signal-` corresponds to the subset of events which pass the respective signal region cuts. `sumofweights2` corresponds to sum of squared weights which is an important quantity to calculate the statistical uncertainty properly in case of weighted events. `normevents` correspond to the physical number of events after normalising to the provided cross section and the analysis' respective integrated luminosity.

`TotalResultFileColumns` and `BestSignalRegionResultFileColumns` can in addition use the following columns:

`obs, bkg, bkgerr, eff, eff_err_stat, eff_err_sys, eff_err_tot, s95obs, s95exp, robs, robscons, robsconssyonly, rexp, rexpcons, rexpconssyonly, clsobs, clsobs_err, clsexp, clsexp_err, likelihood,`

which are mostly self-explanatory. The suffix `-cons` refers to the *r* value in eq. (1) and columns without this suffix do not include the conservative subtraction term of $1.64 \cdot \Delta S$ in the numerator. Note that the output of `CLS` and likelihood related columns are set to `-1` unless the calculation of the respective quantity is enabled via the corresponding flag; see Table 2.

Parameter card	Terminal	Description
General options		
Name: X	-n X	Gives name X to the run and specifies output directory.
Analyses: X	-a X	States which analysis/es X should be applied to the event files; see the text for more details.
SLHAFile: X	-slha X	Use SLHA file X. Mandatory if event generation using Pythia 8 is requested.
InvisiblePIDs: X	-invpids X	BSM Monte Carlo Particle IDs [55] which are invisible for the detector; see the text.
QuietMode: True	-q	No terminal output is produced. Automatically sets -sp.
SkipParamCheck: True	-sp	Skips startup parameter check.
SkipAnalysis: True	-sa	Skips analysis step. Requires -wp8 or -wd.
SkipPythia: True	-spy	Only if .lhe files are provided. These are not showered by Pythia 8 but instead directly processed via Delphes.*
SkipEvaluation: True	-se	Skips evaluation step.
RandomSeed: X	-rs X	Chooses fixed seed X for the random number generator to render output deterministic.
Options related to output		
WritePythiaEvents: True	-wp8	Writes .hepmc files produced by Pythia 8 on disk.
WriteDelphesEvents: True	-wd	Writes .root files produced by Delphes on disk.
EventResult- FileColumns: X	-erfc X	Sets columns which are stored in event-wise result files; see the text for more details.
ProcessResult- FileColumns: X	-prfc X	Sets columns which are stored in process-wise result files; see the text for more details.
TotalResult- FileColumns: X	-tefc X	Sets columns in TotalResults.txt after evaluation; see the text for more details.
BestPerAnalysisResult- FileColumns: X	-bpaefc X	Sets columns in BestPerAnalysis.txt after evaluation; see the text for more details.
OutputDirectory: X	-od X	Specifies directory in which the results should be stored.
OutputExists: X	-oe X	Specifies what to do if output directory already exists. overwrite will delete existing output and overwrite it with the new results. add will add the current results to the old ones, see text.

Table 1: Summary of all parameters which can be set within **CheckMATE**, either via the parameter card introduced in Section 4 or as command line input `./CheckMATE -X -Y . . .`. Occasional dash symbols (-) in the first column only indicate that a command is split in two lines to reduce the column width and are *not* part of the keyword. **The use of LHE events that have only been generated at the parton level or showered events that have been pre-clustered is not recommended and may lead to substantial efficiency and acceptance errors.*

Parameter card	Terminal	Description
Options related to statistical evaluation		
FullCLs: True	-cls	Evaluate full observed and expected CLs for all signal regions and use it for exclusion test.
BestCLs: X	-bcls X	As above but only for the <i>X</i> signal regions with highest rexpcons value.
Likelihood: True	-likeli	Evaluate likelihood for all signal regions and sum the individual contributions; see the text.
EffTab: B	-eff_tab	Creates efficiency tables for every signal region in each analysis run.
No_MC_Stat_Err: True	-mcstats_off	Uncertainty associated with limited Monte Carlo signal statistics is not included in any statistical evaluation.
Process dependent options		
[X]	-p X	Sets up a process with name <i>X</i> .
MaxEvents: X	-maxev X	Defines the number of generated events if Pythia8 or MadGraph5_aMC@NLO is used for event generation. If events are provided, simulation stops when either the end of the event file or MaxEvents is reached.
XSect: X	-xs X	Sets the cross section for the given process including unit (e.g. '10.7 fb'. Note that the space between the value and the unit is required). <i>Must</i> be provided for .hepmc and .root input and <i>can</i> be provided to override the Pythia8 calculated cross section.
XSectErr: X	-xse X	Sets the systematic cross section error for the given process including unit which can be given with a unit or provided as a percentage (e.g. '2.2 fb' or '20.0 %'. Note that the space between the value and the unit is required).
KFactor: X	-kf X	Sets the <i>K</i> -factor for the cross section.
Events: X	-ev X	Sets the .hepmc , .hep , .lhe or .root event files which are to be analysed for the given process.
Pythia8Process: X	-pyp X	Specifies the SUSY process to be generated by Pythia8 ; see the text for more details.
Pythia8Card: X	-pyc X	Specifies the .in input card used by Pythia8 . It can also be used to provide settings for showering and hadronisation of LHE files provided externally or generated with MadGraph5_aMC@NLO .
Pythia8Rndm: X	-pyr X	Provides the binary random number generator state (output during previous run) for the Pythia8 run.
MGprocess: X	-mgproc X	Specifies the process card for generating events with MadGraph5_aMC@NLO .
MGcommand: X	-mgcommand X	Specifies the content of the MadGraph5_aMC@NLO process card explicitly.
MGparam: X	-mgparam X	Optional parameter card for MadGraph5_aMC@NLO (e.g. SLHA card).
MGRun: X	-mgrun X	Optional run card for MadGraph5_aMC@NLO .

Table 2: Continuation of Table 1. Occasional dash symbols (-) in the first column only indicate that a command is split in two lines to reduce the column width and are *not* part of the keyword.

add mode: After a **CheckMATE** run is completed, the user might realise that the events which were processed were insufficient. For example, the size of the tested Monte Carlo samples might be too small to find the number S with desired statistical precision. It might also happen that *a posteriori* it becomes apparent that other processes need to be taken into account which were expected to be negligible before the first run. For those cases, **CheckMATE** allows for new results to be added to old ones. To do so, **CheckMATE** has to be set up with the same name and the same output directory as the original run. During the initialisation step the user is then explicitly asked if the new results are supposed to replace or to be added to the existing ones. If the second option is chosen, the original **CheckMATE** settings are restored from the earlier run, all events in the current setup file are processed and properly added to the ones of the first run. This procedure can be repeated as many times as the user wishes. This behaviour can be controlled by the **OutputExists** parameter or using **-oe add** from the command prompt.

Likelihood: Instead of exclusion tests, **CheckMATE** can also be used for model fits; see Ref. [56] for an example. With this option **CheckMATE** can calculate the likelihood ratios for all signal regions to test the compatibility of a given model with the experimental results. A formulae for a simplified LHC profile likelihood ratios without nuisance parameters is given by Eq. (2) in Appendix B.1. **CheckMATE** uses the full version including ΔB and ΔS as nuisance parameters given by Eq. (5) of Section B.1. In the final results, **CheckMATE** also returns the sum of the likelihood ratios over all signal regions but this value should be used with care since no checks for kinematically overlapping regions are considered. The user is advised to check the signal regions of interest and only sum those that are independent. Analysing the dependence of this quantity on model parameters allows one to find best fit points and the corresponding confidence intervals, see e.g. Ref. [55].

Pythia8Process: Due to the popularity of the MSSM, **CheckMATE** also allows one to easily set up generation of SUSY production processes using the **Pythia8Process** keyword. Possible values for this parameter are **p p > X**, with **X** being any of the following:

go go: gluino pair production;
go sq: gluino-squark and gluino-antisquark associated production;
sq sq : squark-antisquark production;
t1 t1 : pair production of the lightest stop;
3gen: pair production of stops and sbottoms;
sq sq: squark pair production;
colsusy: all coloured SUSY pair-production;
ewsusy: pair production of neutralinos, charginos and neutralino-chargino production;
allsusy: all of the above.

Note that here “squark” always corresponds to the squarks of the first two generations. To simulate any other process or any combination of the above, an explicit **Pythia8.in** file has to be provided. For all processes the default parton distribution function [57] is used and the underlying event is switched off.

4. Example: Running CheckMATE and Understanding the Results

To illustrate how the individual steps explained in Section 2 work in practice, we discuss an example **CheckMATE** run. It is designed in such a way that it covers the most common scenarios to provide input data within the current **CheckMATE** version. It also attempts to apply some optional settings to illustrate their meaning. After the example run is completed, we take a closer look at the auxiliary files which are created along the way and what additional information a user can find in these. A tarball containing all files which are used as input in this example run can be downloaded from <http://www.hepforge.org/archive/checkmate/ExampleFilesForCheckMATE2Manual.tar.gz>

4.1. Benchmark Model

Within this section we test a simplified supersymmetric model where the only kinematically accessible particles are the gluino with mass 1.5 TeV, the eight mass-degenerate squarks of the first two generations with mass 1.5 TeV and a 100 GeV stable bino-like neutralino lightest supersymmetric particle (LSP). Here, the gluino is expected to always decay democratically¹³ into same-flavour quark-antiquark pairs of the first two fermion generations and the stable lightest neutralino. Squarks always decay into the associated Standard Model quark and the neutralino. The masses of squarks and gluino are chosen such that neither of the two can directly decay into the other. The important parts of the SLHA file for this model are as follows:

```
point.slha

[...]
Block MASS # Scalar and gaugino mass spectrum
# PDG code mass particle
1000001 1.500000e+03 # downL squark
1000002 1.500000e+03 # upL squark
1000003 1.500000e+03 # strangeL squark
1000004 1.500000e+03 # charmL squark
1000005 5.000000e+03 # bottom1 squark
1000006 5.000000e+03 # top1 squark
2000001 1.500000e+03 # downR squark
2000002 1.500000e+03 # upR squark
2000003 1.500000e+03 # strangeR squark
2000004 1.500000e+03 # charmR squark
2000005 5.000000e+03 # bottom2 squark
2000006 5.000000e+03 # top2 squark
1000011 5.000000e+03 # electronL slepton (and other sfermions)
[...]
1000021 1.500000e+03 # gluino
1000022 1.000000e+02 # neutralino 1
1000023 5.000000e+03 # neutralino 2 (and other neutralino/charginos)
[...]

# PDG Width
DECAY 1000021 3.154880e-02
2.500000e-01 3 -1 1 1000022 # gluino -> up anti-up neutralino1
2.500000e-01 3 -2 2 1000022 # gluino -> down anti-down neutralino1
2.500000e-01 3 -3 3 1000022 # gluino -> strange anti-strange neutralino1
2.500000e-01 3 -4 4 1000022 # gluino -> charm anti-charm neutralino1

DECAY 1000001 2.105978e-01 # downL decays
1 2 1000022 1 # downL -> down neutralino1
DECAY 1000002 2.105978e-01 # upL decays
1 2 1000022 2 # upL -> up neutralino1
DECAY 1000003 2.105978e-01 # strangeL decays
1 2 1000022 3 # strangeL -> strange neutralino1
DECAY 1000004 2.105978e-01 # charmL decays
1 2 1000022 4 # charmL -> charm neutralino1
DECAY 2000001 8.423913e-01 # downR decays
1 2 1000022 1 # downR -> down neutralino1
DECAY 2000002 8.423913e-01 # upR decays
1 2 1000022 2 # upR -> up neutralino1
DECAY 2000003 8.423913e-01 # strangeR decays
1 2 1000022 3 # strangeR -> strange neutralino1
DECAY 2000004 8.423913e-01 # charmR decays
1 2 1000022 4 # charmR -> charm neutralino1
```

¹³For a bino-like LSP, a gluino would actually decay with different branching-ratios into up-like and down-like quarks due to their different quantum numbers. However, phenomenologically these quarks are almost indistinguishable at the LHC so we can safely set all branching ratios equal.

The most relevant production modes for such a model are the 2-body final states $pp \rightarrow \tilde{g}\tilde{g}, \tilde{g}\tilde{q}, \tilde{q}\tilde{q}$ and $\tilde{q}\tilde{q}^*$. In our example run, we use different approaches¹⁴ to generate the events for these processes:

- For squark-antisquark production, $pp \rightarrow \tilde{q}\tilde{q}^*$, we call `MadGraph5_aMC@NLO` internally to do the parton level event generation and subsequently let `Pythia8` do the parton showering. This mode is one example of how to perform the event generation entirely on the fly. In our example we explicitly give the commands for `MadGraph5_aMC@NLO` to simulate the correct final state.
- Similarly, for the final state $\tilde{q}\tilde{q}$ of squark pair production, we also generate events within `CheckMATE`, however this time we entirely rely on `Pythia8` to do both the partonic event generation and parton showering. Here, we setup `Pythia8` via its `.in` settings file.
- For events of type $\tilde{g}\tilde{g}$, we provide partonic `.lhe` files generated with `MadGraph5_aMC@NLO` beforehand and perform the parton showering and hadronisation with `Pythia8` directly within `CheckMATE`.
- Associated gluino-squark production has been performed completely externally and we provide two fully showered `.hepmc` files. The two files contain the same physics process generated with different random seeds such that they contain statistically independent samples. Such a setup with multiple files per process can for example be required when event generation was parallelised on a computing cluster.

4.2. Initialising and Starting *CheckMATE*

We assume that `CheckMATE` has already been properly installed in folder `$CMDIR` including `Pythia` and `MadGraph5_aMC@NLO` functionalities; see also Appendix A. The pre-generated `.lhe` file for the $\tilde{g}\tilde{g}$ and the `.hepmc` file for the $\tilde{g}\tilde{q}$ process are located in `/scratch/files`. To run `CheckMATE`, some mandatory information has to be provided, either via a command line input or via a text-based parameter card. As the former only works for runs with single process, we have to choose the second approach. A (not minimal) working example for the above setup reads as follows:

```
checkmate.parameters.in

[Parameters]
Name: ExampleRun
SLHAFile: /scratch/files/point.slha
Analyses: 8TeV
RandomSeed: 10

[squ_asq]
MGCommand: import model mssm;
           define sq = ul ur sl sr dl dr cl cr;
           define sq~ = ul~ ur~ sl~ sr~ dl~ dr~ cl~ cr~;
           generate p p > sq sq~;
KFactor: 1.96

[squ_squ]
Pythia8Card: /scratch/files/pythiasqusqu.in
MaxEvents: 1000

[glu_glu]
Events: /scratch/events/glu_glu.lhe
XSectErr: 20 %

[glu_sq]
Events: /scratch/events/glu_squ_1.hepmc, /scratch/events/glu_squ_2.hepmc
XSect: 1.90 fb
```

¹⁴The different input modes are only combined for illustrative purpose here. In practice, one would normally use the same tool setup for all the different hadronic SUSY final states.

The general structure of such a file consists of blocks separated by `[]` expressions which contain one or more **Key: Value** pairs.

The first such block, `[Parameters]`, is a special block type which lists general settings for the **CheckMATE** run common to all processes. In our example, we first give our run a specific name **ExampleRun** which specifies the name of the output directory. With a nonzero **RandomSeed** we make the results deterministic as this parameter fixes the sequence of random numbers which is used to e.g. simulate kinematic configurations in the event generation and apply finite efficiencies on final state objects in the detector simulation phase.

We then provide the already explained `.slha` spectrum file which informs **MadGraph5_aMC@NLO** and **Pythia8** about the masses and decay tables of all SUSY particles. This file is common to all processes since obviously the same physics scenario should be considered within one **CheckMATE** run. In our case, the **SLHAFile** is a mandatory parameter as we ask **CheckMATE** to simulate the events internally and therefore not providing this parameter would result in an immediate abort.¹⁵ Additional possible settings which can be changed via the `[Parameters]` block are summarised in Section 3.

Besides the special `[Parameters]` block, any other `[X]` block combines the information for a particular production process **X**, where **X** is a freely chosen identifier. In our particular case, we need four such blocks for all the different production modes we wish to take into account. Within each such process block we have to provide the information that describes the form of the Monte Carlo events for the particular process given.

- We start with the block `[squ_asq]` responsible for $\tilde{q}\tilde{q}^*$ production. With the **MGCommand** keyword, we specify the set of commands to internally call **MadGraph5_aMC@NLO**. For our example, we load the Minimal Supersymmetric Standard Model via the `import model mssm` command, combine all squarks and all antisquarks into single identifiers **sq**, **sq^{*}**, respectively and conveniently setup pair production of all squark-antisquark pairs via `generate p p > sq sq*`.¹⁶ Since we do not specify otherwise, **MadGraph5_aMC@NLO** will be set up so that it simulates 5,000 partonic events for the given process. Because no explicit cross section is provided, **CheckMATE** uses the result from **MadGraph5_aMC@NLO** determined during the generation of the events. We use the optional **KFactor** parameter to specify a *K*-factor which we determined using **NLLFast**. It multiplies the **MadGraph5_aMC@NLO** leading order cross section with a fixed quantity to estimate the cross section at the next-to-leading order plus next-to-leading log accuracy in QCD.
- For the simulation of $\tilde{q}\tilde{q}$ pair production, we set up a second block called `[squ_squ]`. Here, we also want the event generation to be done entirely internally via **Pythia8**, however this time we explicitly provide the `.in` setting file for **Pythia**.

```
pythiasqusqu.in

PDF:pSet = 8 !(CTEQ6L1)

Beams:idA = 2212    ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212    ! second beam, p = 2212, pbar = -2212
Beams:eCM = 8000.

SLHA:file = /scratch/files/point.slha
SUSY:qq2squarksquark = on
SUSY:idVecA = 1000001,1000002,1000003,1000004,2000001,2000002,2000003,2000004
```

The meaning of the individual lines should be self explanatory. The last row specifies the set of squarks which should be taken into account and which we set to all left- and right-chiral squarks of the first

¹⁵If we instead only provided showered `.hepmc` files, this parameter would not be mandatory as no model-dependent information would be required. Note that providing partonic `.lhe` may require an **SLHA** file including the full decay table of all BSM particles, namely if the BSM final state particles are not yet fully decayed and if the decay table is not included in the header part of the `.lhe` event file already.

¹⁶Note that this call will form all possible final state combinations of the product $(u_l \ u_r \ d_l \ \dots) \times (u_l^* \ u_r^* \ d_l^* \ \dots)$, including flavour-nondiagonal pairs, because of different combinations of initial state quarks in protons.

two generations. In principle, any of **Pythia8**'s parameters listed in Ref. [50] can be changed via this file. Note that for this block, we explicitly specify the number of generated events to be 1,000 via the optional parameter **MaxEvents**.

- The third process block **[glu_glu]** sets up the gluino pair production process where we have already generated 1,000 events using **MadGraph5_aMC@NLO**. Here, we simply have to provide a reference to this file via the **Events** keyword. There is no other mandatory parameter in this case. Most importantly, **CheckMATE** uses standard **Pythia** settings for showering and hadronisation, see below, and takes the cross section from the **.lhe** file itself. In this example, we provide the optional parameter **XSectErr** to inform **CheckMATE** about the systematic error it should consider for this process, which we assume to be 20% of the signal cross section. If no such parameter is provided, as we do for the other three processes, the systematic error associated with the signal is set to zero.
- Finally, we provide two fully hadronic **.hepmc** files for associated gluino squark production in the **[glu_squ]** block. We can simply list all available files for a given process in one **Events** command and in the end the results of all files are properly averaged as explained below. If **CheckMATE** is run with showered **.hep** or **.hepmc** files the cross-section *must* be provided explicitly by the user as contrarily to the **.lhe** format, these event formats do not store this information. In our case, the provided cross section is taken from **Pythia8** which we used to simulate the events but we could have instead used **NLLFast** or **Prospino**.

With the above files ready, we can start **CheckMATE** with the following command:¹⁷

```
Terminal
$CMDIR/bin:  ./CheckMATE checkmate_parameters.in
```

CheckMATE then responds with a summary of the used settings for the given run and asks a user for confirmation.

```
Terminal

  /-----\
  | | | | ' \ / _ \ | | / / | \ | | / _ \ | | | | _ | _ | |
  | | _ | | | | _ / ( _ | < | | | | / _ \ | | | | _ | _ / _ /
  \-----\ | | \ _ \ \ _ \ | \ _ \ | | / _ \ \ _ \ | | _ | _ | _ |

The following settings are used:
Analyses:
  cms_1301_4698_WW (WW production only, 8 TeV, 3.5 fb-1)
  cms_1303_2985 (CMS, alpha_T + b-jets)
  [...]
  atlas_conf_2014_056 (Constraint on stop production from ttbar spin correlations)
  atlas_conf_2015_004 (Search for an invisibly decaying Higgs boson produced via vector ...)
E_CM: 8.0
Processes:
  Process Name: squ_asq
  Input KFactor: 1.96
  Associated event files and/or Monte-Carlo generation runs:
    MG5_aMCNLO Events
      - internal identifier: 'squ_asq'
      - command: import model mssm;
                  define sq = ul ur sl sr dl dr cl cr;
                  define sq~ = ul~ ur~ sl~ sr~ dl~ dr~ cl~ cr~;
                  generate p p > sq sq~;
```

¹⁷Within this chapter, gray text denotes input to be entered by a user.

```

Process Name: squ_squ
Associated event files and/or Monte-Carlo generation runs:
  Pythia8 Events
    - internal identifier: 'squ_squ'
    - .in settings file: /scratch/files/pythiasqusqu.in
    - at most 1000 events are generated and analysed

Process Name: glu_glu
Input cross section error: 20.0 %
Associated event files and/or Monte-Carlo generation runs:
  LHE Events
    - internal identifier: 'glu_glu'
    - path to .lhe file: /scratch/events/glu_glu.lhe

Process Name: glu_sq
Input Cross section: 1.9 fb
Associated event files and/or Monte-Carlo generation runs:
  HepMC events
    - internal identifier: 'glu_sq_event1'
    - path to eventfile: /scratch/events/glu_sq_1.hepmc

  HepMC events
    - internal identifier: 'glu_sq_event2'
    - path to eventfile: /scratch/events/glu_sq_2.hepmc

Output Directory:
  $CMDIR/results/ExampleRun
Additional Settings:
  - SLHA file /scratch/files/point.slha will be used for event generation
  - Fixed random seed of 10
Is this correct? (y/n)

```

Here we chose that **CheckMATE** generates events at $\sqrt{s} = 8$ TeV centre-of-mass energy and tests against all implemented ATLAS and CMS analyses for that particular energy. Note how a unique **internal identifier** is given to each process which will help us to associate output files to the corresponding input events later. In the following, when we use an expression *event*, it is to be understood as a placeholder for one such internal identifier.

As soon as we start **CheckMATE** by answering y, it informs us that — since we did not specify otherwise — **MadGraph5_aMC@NLO** is set up to generate 5,000 events for the **squ_asq** process.

```

Terminal
squ_asq:prepare(): Setting number of to-be-generated MC events to 5000.
                    Use the 'maxEvents' Parameter to change this default behaviour.

```

After about five to ten minutes, depending on user's CPU, during which **CheckMATE** continuously updates us with the current status of the analysis chain, it returns the following result:

```

Terminal
Evaluating Results
Test: Calculation of r = signal/(95%CL limit on signal)
Result: Excluded
Result for r: 2.24273341815
Analysis: atlas_1405_7875
SR: SR02_3j

```

We find that after simulating all events, passing them through a detector simulation and performing 40 different analyses, **CheckMATE** concludes that the input parameter point is excluded because in signal region

SR02_3j of analysis `atlas_1405_7875`, see Ref. [58], the number of predicted signal events S exceeds the 95 % upper limit S_{95} when testing the conservative value,

$$r = \frac{(S - 1.64 \cdot \Delta S)}{S_{95}}. \quad (1)$$

This agrees with the result of the experimental collaboration, see Figure 9 of Ref. [58].

For most users, this information would be sufficient for checking exclusion status of a given model. Simply by changing and testing different values of $m_{\tilde{q}}$ and $m_{\tilde{g}}$ given in the `SLHA` file, one easily finds the allowed and excluded regions in parameter space.

4.3. Structure of the Results Folder

We now take a closer look at the additional information that is stored in many files in the results folder. These files may be ignored by a user who simply performs a test for a given parameter point. However, knowing which information can be found in these files can be very helpful if for instance a more detailed breakdown of intermediate results is required or if `CheckMATE` behaves in an unexpected way. Furthermore, analysing these files aids us in understanding how `CheckMATE` internally works.

For our example case, the results folder would be located under `$CMDIR/results/ExampleRun`. It contains the following files and directories:

```
Terminal
$CMDIR/results/ExampleRun: ls
analysis delphes evaluation fritz internal mg5amcatnlo pythia result.txt
```

The file `result.txt` stores exactly the same information as printed on screen at the end of the `CheckMATE` run. The other folders store the respective individual information of the different modules as explained in the introductory section and we discuss them in the same order.

4.3.1. Folder `internal/`

The `internal` folder stores all internally set `CheckMATE` parameters in a `Python` readable format such that *a posteriori* one is capable of reproducing the exact `Python` instance of `CheckMATE` and is required if `CheckMATE` is run in the `add` mode explained in Section 3. In normal use they are of no relevance to a user and therefore we do not further discuss them here. Note that files in this folder are only created if `CheckMATE` finished successfully and did not abort due to an internal error.

4.3.2. Folder `fritz/`

As `FRITZ` is the steering code which runs and calls the respective submodules, we continue our discussion with a content of the folder `fritz`. After running our example it should contain the following files:

```
Terminal
$CMDIR/results/ExampleRun/fritz: ls
fritz_error.log          fritz_glu_sq_event2.log  glu_glu.ini             squ_asq.ini
fritz_glu_glu.log        fritz_squ_asq.log        glu_sq_event1.ini        squ_squ.ini
fritz_glu_sq_event1.log  fritz_squ_squ.log        glu_sq_event2.ini
```

Here, the `fritz_event.log` files contain the runtime output of `FRITZ` which was also printed on-screen while `CheckMATE` was analysing *event*. `fritz_error.log` combines the standard error output of all runs and should hopefully be empty at all times. Note that — besides the `fritz_error.log` file — there exists one `.log` file for each of the individually tested event files. Therefore, in our case, we have *five* files as we have *four processes* including one which uses two separate `.hepmc` event files. The respective `.log` files informs a user about the order in which individual modules were initialised, combined and finalised in the end, for example:

fritz_squ_squ.log

```
Fritz: Initialising handlers from file $CMDIR/results/ExampleRun/fritz/squ_squ.ini
Fritz: Set random seed to 10
PythiaHandler: Output redirected to $CMDIR/results/ExampleRun/pythia/pythia_squ_squ.log
PythiaHandler: Initializing Pythia8 with /scratch/files/pythiasqusqu.in
PythiaHandler 'squ_squ': Pythia8 initialized successfully!
PythiaHandler 'squ_squ': Pythia8 will generate 1000 events
DelphesHandler 'atlas8tev': Initialising Delphes via linking to PythiaHandler 'squ_squ'
DelphesHandler 'atlas8tev': Initialising settings from $CMDIR/results/ExampleRun/delphes/modified_...
DelphesHandler 'atlas8tev': Delphes successfully initialised!
DelphesHandler 'cms8tev': Initialising Delphes via linking to PythiaHandler 'squ_squ'
DelphesHandler 'cms8tev': Initialising settings from $CMDIR/results/ExampleRun/delphes/modified_cm...
DelphesHandler 'cms8tev': Delphes successfully initialised!
AnalysisHandler 'atlas8tev': Initialising AnalysisHandler
AnalysisHandler 'atlas8tev': Loading Analysis atlas_1308_1841
AnalysisHandler 'atlas8tev': Successfully loaded analysis atlas_1308_1841
[...]
AnalysisHandler 'atlas8tev': Successfully loaded analysis atlas_conf_2015_004
AnalysisHandler 'atlas8tev': Linking to DelphesHandler 'atlas8tev' tree
AnalysisHandler 'atlas8tev': AnalysisHandler successfully linked to DelphesHandler 'atlas8tev'
AnalysisHandler 'cms8tev': Initialising AnalysisHandler
AnalysisHandler 'cms8tev': Loading Analysis cms_1301_4698_WW
[...]
AnalysisHandler 'cms8tev': Linking to DelphesHandler 'cms8tev' tree
AnalysisHandler 'cms8tev': AnalysisHandler successfully linked to DelphesHandler 'cms8tev'
Fritz: Fritz successfully loaded command line parameters!
Fritz: >> Successfully initialized and linked all handlers! <<
Fritz: Starting event loop!
Fritz: Progress: 10 %
Fritz: Progress: 20 %
[...]
Fritz: Progress: 100 %
Fritz: >> Finalising after 1000 events. <<
AnalysisHandler 'atlas8tev': Asking DelphesHandler 'atlas8tev' for cross section information
DelphesHandler 'atlas8tev': Asking PythiaHandler 'squ_squ' for cross section information
PythiaHandler 'squ_squ': Pythia8 returned cross section of 2.43366 fb
PythiaHandler 'squ_squ': Pythia8 returned cross section error of 0 fb
AnalysisHandler 'atlas8tev': Analyses updated with sigma = 2.43366 fb and dSigma = 0 fb
AnalysisHandler 'atlas8tev': Analyses successfully finished!
AnalysisHandler 'cms8tev': Asking DelphesHandler 'cms8tev' for cross section information
DelphesHandler 'cms8tev': Asking PythiaHandler 'squ_squ' for cross section information
PythiaHandler 'squ_squ': Pythia8 returned cross section of 2.43366 fb
PythiaHandler 'squ_squ': Pythia8 returned cross section error of 0 fb
AnalysisHandler 'cms8tev': Analyses updated with sigma = 2.43366 fb and dSigma = 0 fb
AnalysisHandler 'cms8tev': Analyses successfully finished!
DelphesHandler 'atlas8tev': Delphes finished successfully!
DelphesHandler 'cms8tev': Delphes finished successfully!
PythiaHandler 'squ_squ': Pythia8 finished successfully!
Fritz: >> Done <<
```

This file enables us to trace exactly which modules have been loaded using which settings and how they were respectively linked. In our particular example, for the squark pair production process we need a `PythiaHandler` which takes care of the event generation within `Pythia8`.¹⁸ Then, since we test against all 8 TeV analyses, we require two separate `Delphes` instances, one for the ATLAS and another one for the CMS detector description. Each of these `DelphesHandlers` takes its event information from the same single `PythiaHandler`, i.e. the same generated events are used for both ATLAS and CMS. The detector level objects are then respectively passed to two individual `AnalysisHandlers` which perform flavour tagging and final state isolation checks according to the requirements of the respective analyses. These then pass the information to all the loaded analyses which independently perform the signal region categorisation. In

¹⁸Note that for practical purposes, each process starts a separate FRITZ run. This is why only one `Pythia8` instance appears in the above example logfile for the gluino run.

the end, the cross section information — which is needed by the individual analyses to properly normalise their final results — in this particular case is taken from `Pythia8` itself. Note that the cross section error from the event generator is set to zero, even though this value — as can be seen in the respective output files showed below — has a numerical uncertainty. This *statistical* uncertainty is however already accounted for by `CheckMATE` internally. Any additional *systematic* uncertainty has to be provided by the user via the `XSectErr` keyword.

The required set of handlers, their properties and how they are linked is determined by the `Python` part of `CheckMATE` and for each *event* is passed via an *event.ini* file, for example:

```
glu_glu.ini

[Global]
randomseed = 10

[...]

[PythiaHandler: glu_glu]
pythiapath = $CMDIR/results/ExampleRun/pythia
logfile = $CMDIR/results/ExampleRun/pythia/pythia_glu_glu.log
usemg5 = false
settings = $CMDIR/results/ExampleRun/pythia/glu_glucard_0.in
xsecterrfactor = 0.2

[DelphesHandler: cms8tev]
settings = $CMDIR/results/ExampleRun/delphes/modified_cms8tev_card.tcl
logfile = $CMDIR/results/ExampleRun/delphes/delphes_glu_glu.log
pythiahandler = glu_glu

[DelphesHandler: atlas8tev]
settings = $CMDIR/results/ExampleRun/delphes/modified_atlas8tev_card.tcl
logfile = $CMDIR/results/ExampleRun/delphes/delphes_glu_glu.log
pythiahandler = glu_glu

[AnalysisHandler: atlas8tev]
analysistype = atlas8tev
outputprefix = gluinos
outputdirectory = $CMDIR/results/ExampleRun/analysis
logfile = $CMDIR/results/ExampleRun/analysis/output
delpheshandler = atlas8tev

[AnalysisHandler: cms8tev]
analysistype = cms8tev
outputprefix = gluinos
outputdirectory = $CMDIR/results/ExampleRun/analysis
logfile = $CMDIR/results/ExampleRun/analysis/output
delpheshandler = cms8tev
```

The two `AnalysisHandlers` responsible for ATLAS and CMS analyses need to apply flavour tagging and isolation conditions after the detector simulation step. As explained before, `CheckMATE` first analyses the respective analysis implementation in order to create a list of all required settings for each analysis. All this is stored in the same *.ini* file:

```
glu_glu.ini

[ANALYSIS: atlas_1308_1841]
analysishandler = atlas8tev
jet_btags = atlas8tev7 atlas8tev8 atlas8tev9
electron_isolation = atlas8tev0 atlas8tev3 atlas8tev4
muon_isolation = atlas8tev0 atlas8tev11 atlas8tev12
photon_isolation = atlas8tev0
```

```

[ANALYSIS: atlas_1308_2631]
analysishandler = atlas8tev
jet_btags = atlas8tev5
electron_isolation = atlas8tev0 atlas8tev7
muon_isolation = atlas8tev0 atlas8tev9
photon_isolation = atlas8tev0 atlas8tev0

[...]

[ANALYSIS: atlas_conf_2015_004]
analysishandler = atlas8tev
jet_btags = atlas8tev3
electron_isolation = atlas8tev0 atlas8tev6
muon_isolation = atlas8tev0 atlas8tev6
photon_isolation = atlas8tev0

[BTAG: atlas8tev0]
eff = 70.
analysishandler = atlas8tev

[...]

[BTAG: atlas8tev13]
eff = 75.
analysishandler = atlas8tev

[TAUTAG: atlas8tev0]
analysishandler = atlas8tev

[ELECTRONISO: atlas8tev0]
source = c
analysishandler = atlas8tev
dr = 0.2
ptmin = 0.1
absorrel = r
maxval = 0.2

[ELECTRONISO: atlas8tev1]
source = t
analysishandler = atlas8tev
dr = 0.2
ptmin = 1.
absorrel = r
maxval = 0.1

[.]

[ELECTRONISO: atlas8tev30]
source = c
analysishandler = atlas8tev
dr = 0.3
ptmin = 0.1
absorrel = r
maxval = 0.14

```

```

[MUONISO: atlas8tev0]
source = t
analysishandler = atlas8tev
dr = 0.05
ptmin = 0.5
absorrel = r
maxval = 0.2

[MUONISO: atlas8tev1]
source = t
analysishandler = atlas8tev
dr = 0.2
ptmin = 0.5
absorrel = a
maxval = 1.8

[...]

[MUONISO: atlas8tev27]
source = t
analysishandler = atlas8tev
dr = 0.2
ptmin = 0.4
absorrel = r
maxval = 0.10

[PHOTONISO: atlas8tev0]
source = c
analysishandler = atlas8tev
dr = 0.2
ptmin = 0.1
absorrel = r
maxval = 0.2

[PHOTONISO: atlas8tev1]
source = c
analysishandler = atlas8tev
dr = 0.4
ptmin = 0.1
absorrel = a
maxval = 4.0

[ANALYSIS: cms_1303_2985]
analysishandler = cms8tev
jet_btags = cms8tev3
electron_isolation = cms8tev0 cms8tev8
muon_isolation = cms8tev0 cms8tev9
photon_isolation = cms8tev0 cms8tev1

[...]

[PHOTONISO: cms8tev1]
source = c
analysishandler = cms8tev
dr = 0.3
ptmin = 0.1
absorrel = r
maxval = 0.2
[...]
```

We find, for example, that the 31 ATLAS analyses (including those that are only partially validated, cf. Section 5) of our general 8 TeV run require in total 30 distinct isolation tests for electrons, 27 different muon isolation tests and 13 different b -tagging working points.

The flexibility of FRITZ becomes apparent when inspecting the corresponding .log file for e.g. the $\tilde{g}\tilde{q}$ process for which we provided the fully hadronised .hepmc files:

```
fritz.glu_sq_event1.log

Fritz: Initialising handlers from file $CMDIR/results/ExampleRun/fritz/glu_sq_event1.ini
Fritz: Set random seed to 10
DelphesHandler 'atlas8tev': Initialising Delphes via input event /scratch/events/glu_squ_1.hepmc
DelphesHandler 'atlas8tev': Input File determined to be HepMC.
DelphesHandler 'atlas8tev': Initialising settings from $CMDIR/results/ExampleRun/delphes/modified...
DelphesHandler 'atlas8tev': Delphes successfully initialised!
DelphesHandler 'atlas8tev': Input file successfully opened!
[...]

Fritz: >> Finalising after 1000 events. <<
AnalysisHandler 'atlas8tev': Asking DelphesHandler 'atlas8tev' for cross section information
DelphesHandler 'atlas8tev': Asking EventFile 'glu_sq_event1' for cross section information
EventFile 'glu_sq_event1': Returning cross section of 1.9 fb
EventFile 'glu_sq_event1': Returning cross section error of 0 fb
AnalysisHandler 'atlas8tev': Analyses successfully Finished!
[...]

Fritz: >> Done <<
```

One finds that no PythiaHandler is loaded in this case. The .hepmc files are directly loaded into Delphes and the cross section is taken from a user which is passed to FRITZ and then to the analyses via the corresponding .ini file:

```
glu_sq_event1.ini

[...]

[EventFile: glu_sq_event1]
file = /scratch/events/glu_squ_1.hepmc
xsct = 1.9
xscterr = 0

[...]
```

The analysis part, however, is the same for all processes.

4.3.3. Folder *mg5amcatnlo/*

Results of the event generation by MadGraph5_aMC@NLO are stored in this folder. We get the following files and directories:

```
Terminal

$CMDIR/results/ExampleRun/mg5amcatnlo/: ls
mg5amcatnlo_squ_asq.log  squ_asq  squ_asq_proc_card.dat  squ_asq_run_card.dat
```

The names of all files and directories can be associated to the process that was generated via the respective unique identifier. In our case, we only encounter the identifier **squ_asq** as we enabled internal event generation with MadGraph5_aMC@NLO only for this process.

A typical MadGraph5_aMC@NLO run requires three files:

- A **proc** card lists the commands which should be given to MadGraph5_aMC@NLO to simulate the correct events. In our case, the file **squ_asq_proc_card.dat** contains the exact command we provided via the MGCommand parameter in our initial CheckMATE input file.
- A **param** card typically defines the parameters of the to-be-analysed BSM model in the form of an SLHA file. In our case, this file is already existing and was defined by the SLHA command. It, therefore, does not appear as a separate file in this folder.
- A **run** card specifies the details of the partonic Monte Carlo simulation. This file is created by CheckMATE, taking a standardised run card and filling it with information given by a user, e.g. the number of requested events and the centre-of-mass energy.

squ_asq_run_card.dat

```
#####
#                               MadGraph5_aMCNLO                               *
#                               *                                              *
#                               run_card.dat MadEvent                        *
#                               *                                              *
# This file is used to set the parameters of the run.                        *
#                               *                                              *
# Some notation/conventions:                                                *
#                               *                                              *
# Lines starting with a '#' are info or comments                          *
#                               *                                              *
# mind the format:  value      = variable      ! comment                    *
#####
#
#*****
# Running parameters
#*****
#
[...]
5000 = nevents ! Number of unweighted events requested
10   = iseed   ! rnd seed (0=assigned automatically=default))
[...]
1     = lpp1    ! beam 1 type
1     = lpp2    ! beam 2 type
4000.0 = ebeam1 ! beam 1 total energy in GeV
4000.0 = ebeam2 ! beam 2 total energy in GeV
[...]
```

The `squ_asq` folder is the working directory of `MadGraph5_aMC@NLO` which contains all input files, executables and output files. For more information regarding the meaning of these we refer to the original publication, Ref. [59].

4.3.4. Folder *pythia*/

The `pythia` folder contains all files that have been used to simulate the events with `Pythia8`. In our particular case, these are

Terminal

```
$CWD/~/results/ExampleRun/pythia: ls
glu_glucard_0.in  pythia_squ_asq.log  rndm-end.dat  squ_asq_showercard.in
pythia_glu_glu.log  pythia_squ_squ.log  rndm-init.dat
```

The `.in` files correspond to setup files for `Pythia8` which have been created by `CheckMATE` and which contain the commands to simulate the respective final state with the correct centre-of-mass energy using the SUSY parameter point from the `.slha` file. We have such a card for the `[squ_asq]` process, for which we used the `MGCommand` parameter. Here, `CheckMATE` generates the `.in` file used for showering the parton events from `MadGraph5_aMC@NLO` automatically.

squ_asq_showercard.in

```
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = on ! list changed particle data
Main:timesAllowErrors = 300        ! how many aborts before run stops
PartonLevel:MPI = off              ! no multiparton interactions
SLHA:file = /scratch/files/point.slha
```

For the $\tilde{g}\tilde{g}$ production, we also have a card which sets up `Pythia8` for hadronisation of the events:

```

glu_glucard.in
[...]
```

Next:numberShowEvent = 0	! print event record n times
--------------------------	------------------------------

```

Beams:frameType = 4
Beams:LHEF = /scratch/events/glu_glu.lhe
```

The `.log` files contain the verbatim output produced by `Pythia8` before, during and after the event generation. If the simulation finished successfully, this file concludes with a summary of the numerically evaluated cross sections, for example:

```

pythia_squ_squ.log

[...]
```

----- PYTHIA Event and Cross Section Statistics -----						
Subprocess	de	Number of events			sigma +- delta	
		Tried	Selected	Accepted	(estimated)	(mb)

q q' -> ~d_L ~d_L + c.c.	1351	625	12	12	4.536e-14	4.978e-15
q q' -> ~d_L ~s_L + c.c.	1352	22	0	0	0.000e+00	0.000e+00
q q' -> ~d_L ~b_1 + c.c.	1353	0	0	0	0.000e+00	0.000e+00
[...]						
q q' -> ~t_2 ~d_L + c.c.	1423	0	0	0	0.000e+00	0.000e+00
q q' -> ~t_2 ~s_L + c.c.	1424	0	0	0	0.000e+00	0.000e+00

sum		27950	1000	1000	2.367e-12	3.489e-14

----- End PYTHIA Event and Cross Section Statistics -----						
[...]						

Note that, as expected, the final numbers coincide with the cross section values quoted in the `fritz_squ_squ.log` file that have been used for the analysis normalisation. If, for some reason, the event generation has to be aborted, this file can contain more information about the cause.

The run also produces two binary files `rndm-init.dat` and `rndm-end.dat` which contain the state of the random number generator. The random number sequence can be reproduced by providing `Pythia8Rndm:<path>/rndm-init.dat` in the parameter file. Alternatively, providing `rndm-end.dat` ensures that the new random number sequence is independent. If a user requires the directory to be overwritten, these files should be first copied to another location.

4.3.5. Folder *delphes/*

Analogously to the `pythia` folder, intermediate results of the detector simulation step are stored in the `delphes` folder.

```

Terminal
$CMDIR/results/ExampleRun/delphes: ls
delphes_glu_glu.log      delphes_squ_asq.log      modified_cms8tev_card.tcl
delphes_glu_sq_event1.log delphes_squ_squ.log
delphes_glu_sq_event2.log modified_atlas8tev_card.tcl
```

For our example, this folder simply contains the log files produced by `Delphes` for each of the five independent event files. These do not contain any interesting information if a run succeeded but can be of assistance if the detector simulation encountered an unexpected problem. Typically, `CheckMATE` uses standardised `.tcl` detector cards which can be found in `data/cards` within the `CheckMATE` installation directory. Only in our specific case, since we fix the random seed which can only be done in `Delphes` in the `.tcl` files, `CheckMATE` produces `modified*.tcl` cards. The only difference to the standard cards is the appearance of the line `set RandomSeed 10` at the very end. For more information about the fast detector simulation `Delphes`, we refer to Ref. [1].

4.3.6. Folder *analysis*/

A closer look into the **analysis** folder reveals a plethora of files.

```
terminal
$CMDIR/results/ExampleRun/analysis: ls
glu_glu_atlas_1308_1841_cutflow.dat      glu_sq_event2_atlas_conf_2013_024_signal.dat
glu_glu_atlas_1308_1841_signal.dat      glu_sq_event2_atlas_conf_2013_031_cutflow.dat
[...]
glu_sq_event2_atlas_conf_2012_147_signal.dat  squ_squ_cms_sus_13_016_cutflow.dat
glu_sq_event2_atlas_conf_2013_021_signal.dat  squ_squ_cms_sus_13_016_signal.dat
glu_sq_event2_atlas_conf_2013_024_cutflow.dat
```

To be precise, there are two files for each of the analysis for each process event file which in our case sums up to almost 400 different files. The content of these files has not changed since the original **CheckMATE** publication in Ref. [26]. For completeness, we discuss the logic and content of these files here.

Each analysis in **CheckMATE** produces two types of output: **cutflow**-files show the absolute and relative numbers of events that pass the individual selection cuts of the corresponding analysis step-by-step, whereas **signal**-files give the final number of events that pass all signal region cuts defined within the analysis. As shown below, both files have a common structure. For a detailed discussion we choose the analysis **atlas_1405-7875** which **CheckMATE** determined to be responsible for the signal exclusion:

```
glu_glu_atlas_1405.7875_cutflow.dat

# ATLAS
# ATLAS-1405-7875
# 0 lepton, 2-6 jets, etmiss
# sqrt(s) = 8 TeV
# int(L) = 20.3 fb^-1

Inputfile:
XSect:      0.0591437 fb
Error:      0.0118287 fb
MCEvents:   1000
SumOfWeights: 1000
SumOfWeights2: 1000
NormEvents: 1.19582

Cut          Sum_W  Sum_W2  Acc   N_Norm
a.2jl_CR01_all      1000   1000    1     1.19582
a.2jl_CR02_missETjetsPT  940    940   0.94   1.12407
a.2jl_CR07_dphiMin2J3J  762    762   0.762  0.911212
a.2jl_CR11_RHT      616    616   0.616  0.736622
[...]
o.6jt+_CR12_Rmeff    146    146   0.146  0.174589
o.6jt+_CR13_meffIncl 139    139   0.139  0.166218
```

```
glu_glu_atlas_1405.7875_signal.dat

[...]
# ATLAS
# ATLAS-1405-7875
# 0 lepton, 2-6 jets, etmiss
# sqrt(s) = 8 TeV
# int(L) = 20.3 fb^-1

Inputfile:
XSect:      0.0591437 fb
Error:      0.0118287 fb
MCEvents:   1000
SumOfWeights: 1000
SumOfWeights2: 1000
NormEvents: 1.19582
```

SR	Sum_W	Sum_W2	Acc	N_Norm
SR01_a.2jl	616	616	0.616	0.736622
SR01_b.2jm	326	326	0.326	0.389836
SR01_c.2jt	314	314	0.314	0.375486
SR01_d.2jW	27	27	0.027	0.032287
SR02_3j	237	237	0.237	0.283408
SR03_a.4jl-	432	432	0.432	0.516592
SR03_b.4jl	432	432	0.432	0.516592
SR03_c.4jm	50	50	0.05	0.0597908
SR03_d.4jt	232	232	0.232	0.277429
SR03_e.4jW	7	7	0.007	0.00837071
SR04_5j	269	269	0.269	0.321674
SR05_a.6jl	119	119	0.119	0.142302
SR05_b.6jm	119	119	0.119	0.142302
SR05_c.6jt	82	82	0.082	0.0980569
SR05_d.6jt+	139	139	0.139	0.166218

These files start with some general information about the analysis and the analysed events. Note that the cross section error corresponds to 20 % of the total cross section as specified in our **CheckMATE** input file for the $\tilde{g}\tilde{g}$ process.

After this, a list of all individual cutflow milestones/signal regions follows. For each of these, **CheckMATE** lists the sum of weights and sum of squared weights of all events that passed the corresponding cut(s) (**Sum_W**, **Sum_W2**), the relative efficiency times acceptance factor (**Acc**) as well as the expected number of events after normalising to the given total cross section and the luminosity of the respective analysis (**N_Norm**). In case of unweighted events, **Sum_W** and **Sum_W2** corresponds to the number of Monte Carlo events in the respective region and this is true for the example above. However, if weighted events are used, they are properly taken into account and both **Sum_W** and **Sum_W2** are required by **CheckMATE**'s evaluation routines to properly calculate the statistical error in the upcoming evaluation step.

The cutflow information, similarly to all the files discussed in the previous paragraph, can be used e.g. for validation purposes. It is, however, currently not further processed by **CheckMATE**. The **signal** files, on the other hand, contain crucial information used for the subsequent evaluation step explained below.

Any output or warning/error messages generated during the analysis runs are stored in **analysisstdout** *_analysisname.log* files. If after a successful **CheckMATE** run these files are empty — as usually expected — they are removed automatically.

4.3.7. Folder evaluation/

The **evaluation** folder of our example run contains the following files:

Terminal		
\$CMTDIR/results/ExampleRun/evaluation: ls		
best_signal_regions.txt	glu_sq_event2_eventsResults.txt	squ_squ_eventsResults.txt
glu_glu_eventsResults.txt	glu_sq_processResults.txt	squ_squ_processResults.txt
glu_glu_processResults.txt	squ_asq_eventsResults.txt	total_results.txt
glu_sq_event1_eventsResults.txt	squ_asq_processResults.txt	

Files with name **X.eventResults.txt** collect the results returned by all signal regions in all analyses for a given process **X**. By default, the data stored are the normalised number of predicted signal events (**signal_normevents**) and the total error on this number (**signal_err_tot**).

glu_glu_eventResults.txt			
analysis	sr	signal_normevents	signal_err_tot
atlas_1308_1841	SR01_8j50_a.0b	0.0263608	0.00770593751115
atlas_1308_1841	SR01_8j50_b.1b	0.00838752	0.00358665176569
[...]			

atlas_1404_2500	SR3Llow	0.0	0
atlas_1404_2500	SR3b	0.0	0
atlas_1405_7875	SR01_a.2jl	0.736622	0.150283717126
atlas_1405_7875	SR01_b.2jm	0.389836	0.080901268657
[...]			
cms_sus_13_016	SR1	0.0	0

The error is defined as the quadratic sum of the statistical error, calculated internally from the size of the Monte Carlo sample, and the systematic error provided by the user. These individual error sources and additional columns can be requested by setting the correct options in the `[Parameters]` block in the `CheckMATE` setup file; see Section 3.

During the evaluation phase, results from all individual processes are combined. First, results that correspond to the same process will be *averaged* by taking the corresponding weights properly into account. The statistical error is then calculated from the combined sum of weights and combined sum of squared weights. The statistical error for all signal regions with 0 Monte Carlo events at this stage is set to the corresponding statistical error of 1 Monte Carlo event.¹⁹ The combined results of this procedure are stored in `X_processResults.txt`.

glu_sq_event1_eventResults.txt			
[...]			
atlas_1405_7875	SR02_3j	10.9686	0.459030034105
[...]			
glu_sq_event2_eventResults.txt			
[...]			
atlas_1405_7875	SR02_3j	11.2566	0.657620946569
[...]			
glu_sq_processResults.txt			
[...]			
atlas_1405_7875	SR02_3j	11.0645722039	0.376429600571
[...]			

For processes with only one event file, the corresponding `eventResults` and `processResults` files are almost identical, except for the statistical error of signal regions with 0 events which is only set process-wise, not event-wise.

For the next step, results from different processes are *added* to determine the total expected number of signal events for each signal region. All errors are considered independent and hence added in quadrature. This is done for each signal region in each selected analysis separately. These results are then compared to the experimental limits using the chosen method, in our case the conservative *r*-limit since we did not specify anything else. The results for each analysis and each signal region are then stored in `total_results.txt`. Here, the standard columns are the number of experimentally observed, *o*, and expected Standard Model events, *b* ± *db*, quoted by the experiments, the `CheckMATE` predicted number of signal events, *s*, and the corresponding error, *ds*, the model independent 95 % observed and expected limits, *s95obs* and *s95exp*, and the conservative *r* value as defined in Eq. (1).

¹⁹This prescription ensures that Monte Carlo samples for processes with very large cross sections but with an insufficiently small number of events contribute with large statistical uncertainty to the final number, even if no signal event passed the cuts.

total_results.txt										
analysis	sr	o	b	db	s	ds	s95obs	s95exp	robscons	rexpcons
atlas_1308_1841	SR01_8j50_a.0b	40.0	35.0	4.0	0.155	0.063	20.0	16.0	0.002543	0.003178
atlas_1308_1841	SR01_8j50_b.1b	44.0	40.0	10.0	0.021	0.050	23.0	23.0	0	0
atlas_1308_1841	SR01_8j50_c.GE2b	44.0	50.0	10.0	0.028	0.051	22.0	26.0	0	0
atlas_1308_1841	SR02_9j50_a.0b	5.0	3.3	0.7	0.048	0.053	7.0	5.0	0	0
[...]										
atlas_1405_7875	SR01_d.2jW	0.0	2.3	1.4	2.482	1.286	4.8	4.0	0.420647	0.504776
atlas_1405_7875	SR02_3j	7.0	5.0	1.2	19.58	1.222	8.2	6.4	2.242733	2.873502
atlas_1405_7875	SR03_a.4j1-	2169.	2120	110.	22.54	0.282	270.0	240.0	00.07914	0.089036
[...]										
cms_sus_13_016	SR1	1.0	1.2	1.04	0.0	0.048	4.0	3.9	0	0

(Note that we rounded the numbers in the table compared to the actual file content to fit the page width.)

In the last step of the evaluation procedure, **CheckMATE** will search for the signal region with the largest expected sensitivity. For the r -limits this corresponds to the signal region with the largest **rexpcons**. The results of the most sensitive signal region of each analysis is written in the file **best_signal_regions.txt**.

best_signal_regions.txt									
analysis	sr	b	db	s	ds	s95obs	s95exp	robscons	rexpcons
atlas_1308_1841	SR04_7j80_a.0b	11.0	2.2	0.21	0.0685	10.0	10.0	0.010426	0.0104267
atlas_1308_2631	SRA3	15.8	2.8	0.0	0.0492	9.0	10.2	0	0
atlas_1402_7029	SR0taua20	0.29	0.18	0.0	0.0497	2.9	2.9	0	0
atlas_1403_4853	L110	9.3	3.5	0.0	0.0497	9.0	9.4	0	0
atlas_1403_5222	SR2B	2.4	0.9	0.0	0.0497	3.4	4.5	0	0
atlas_1403_5294	WWa_DF	73.6	7.9	0.0	0.0497	20.3	22.533	0	0
atlas_1403_5294_CR	CRmT2_top	789.	126.0	0.0	0.0497	253.0	250.0	0	0
atlas_1404_2500	SR1b	4.7	2.1	0.0	0.0497	13.3	8.0	0	0
atlas_1405_7875	SR02_3j	5.0	1.2	19.5	0.7296	8.2	6.4	2.242733	2.8735021
atlas_1407_0583	bCd_high1	11.0	1.5	0.0	0.0497	13.2	8.5	0	0
atlas_1407_0600	SR017jA	21.2	4.6	0.01	0.0492	13.9	13.8	0	0
atlas_1407_0608	M3	1770	81.0	24.4	1.0220	195.0	190.0	0.116903	0.1199802
atlas_1411_1559	SRTotal	557.	45.0	0.21	0.0880	70.0	91.0	0.000980	0.0007540
atlas_1501_07110	SRmm-1	3.8	0.9	0.0	0.0497	7.9	6.0	0	0
atlas_1502_01518	SR9	97.0	14.0	24.4	0.9640	58.0	36.0	0.394622	0.6357800
atlas_1503_03290	SR-Z	10.6	3.2	0.0	0.0497	29.6	12.0	0	0
atlas_1506_08616	SRinB	14.1	2.8	0.0	0.0497	16.1	11.2	0	0
atlas_conf_2012_10	e1	9.0	2.8	0.0	0.0142	9.9	9.3	0	0
atlas_conf_2012_14	4	380.	73.40	4.47	0.3204	210.0	210.0	0.018805	0.0188058
atlas_conf_2013_02	emumu	287.	19.0	0.0	0.0318	58.2	49.7	0	0
atlas_conf_2013_02	SR1	17.5	3.2	0.07	0.0511	10.0	10.6	0	0
atlas_conf_2013_03	Higgs	3450	180.0	0.0	0.0507	484.0	363.0	0	0
atlas_conf_2013_03	SR1Z	1.3	1.0	0.0	0.0507	6.5	4.5	0	0
atlas_conf_2013_04	SR_mT2_110_elmu	4.4	2.0	0.0	0.0497	7.105	6.699	0	0
atlas_conf_2013_06	SR0L7JA	22.5	6.9	0.01	0.0492	15.3	14.6	0	0
atlas_conf_2013_06	SoftLep1BHigh	4.0	1.1	0.02	0.0508	7.9	6.3	0	0
atlas_conf_2013_08	SR10F	103.	15.0	0.0	0.0497	24.0	31.0	0	0
atlas_conf_2014_01	SRA	53.0	10.0	0.0	0.0497	30.2	27.0	0	0
atlas_conf_2014_03	emu	4376	281.2	0.0	0.0497	1176.0	566.0	0	0
atlas_conf_2014_05	sig	6000	3600.	0.0	0.0497	6902.0	6717.0	0	0
atlas_conf_2015_00	M1	578.	48.41	0.04	0.0497	73.0	96.0	0	0
cms_1301_4698_WW	combined	1000	60.0	0.0	0.00855	240.4	135.7	0	0
cms_1303_2985	23j_0b_875	16.1	1.7	7.75	0.4298	18.545	10.11550	0.379938	0.6965528
cms_1405_7570	Zjj_030	2136	859.0	0.0	0.0477	1378.8	1595.450	0	0
cms_1408_3583	550	509.	66.0	9.28	0.6481	129.0	123.0	0.063758	0.0668690
cms_1502_06031	SR01_GE2jets_c.highMET	12.8	4.3	0.0	0.0475	7.6	7.6	0	0
cms_1504_03198	SR1	16.4	3.640	0.0	0.0482	12.9	11.4	0	0
cms_smp_12_006	0e	487.	40.0	0.0	0.0480	151.62	88.98	0	0
cms_sus_12_019	For_OF	155.	16.40	0.0	0.0475	31.8	31.8	0	0
cms_sus_13_016	SR1	1.2	1.048	0.0	0.0477	4.0	3.9	0	0

This file is helpful in getting a good overview of which analyses yield a non-vanishing r value and hence show sensitivity to the tested model. In our example, one would expect that the most sensitive analyses

are those targeting final states with a large jet multiplicity and missing transverse energy. Indeed, one can identify three such analyses with sizeable r -values: `atlas_1405.7875` [58], the zero lepton multijet search, the ATLAS monojet²⁰ search `atlas_1502.01518` [60] and the CMS search `cms_1303.2985` [61] which uses the α_T variable to identify BSM events with a large hadronic activity.

CheckMATE then again chooses the most sensitive signal region among these. The corresponding *observed* result will be used to finally conclude whether the input can be considered excluded or not, i.e. in the case of the r -limit if `robscons` is larger than 1. In the above example, the best signal region would be `SR02.3j` in the analysis `atlas_1405.7875` which with the `robscons` value of about 2.2 excludes the tested model. This is exactly the result which was printed on screen at the end of our **CheckMATE** run.

With that, we have illustrated how **CheckMATE** can be used to test various BSM models with a range of input methods and which content can be found in all the produced output files. This knowledge should be sufficient for standard users to test their models of interest without much effort.

5. Available Analyses

A large number of ATLAS and CMS studies have been implemented covering a wide range of final state configurations. The vast majority of studies are dedicated searches for supersymmetry and generally require significant missing transverse momentum. They can be divided into four main categories.

- The powerful inclusive SUSY searches in final states with large jet multiplicities and large missing transverse momentum target the production of gluino and first/second generation squark pairs which subsequently cascade or directly decay into the lightest stable supersymmetric particle.
- Third generation searches are of particular interests due to the emerging little hierarchy problem and are sensitive to direct production of stop and sbottom pairs. In addition, searches for gluino induced stop and sbottom production (in cascade decays) are implemented in **CheckMATE**. The decays of the third generation sparticles yield top or bottom quarks in the final state.
- Electroweak searches target the direct production of electroweakinos and sleptons and generally focus on final states with at least two leptons (electrons or muons) or taus.
- Monojet (monophoton) searches which are sensitive to compressed spectra, simplified dark matter models, large extra dimensions, Higgs portals and other scenarios predicting a high momentum jet (photon) recoiling against missing transverse momentum.

A search for production of vector like top quarks as well as the rapidity gap signature in the vector boson fusion have also been implemented. Several analyses focused on SM measurements, like cross sections, are available as well. Searches for long-lived particles as well as the heavy Higgs boson searches are not included in the current **CheckMATE** version. Even though most of the currently implemented searches focus on SUSY, they can be applied to *any* non-supersymmetric BSM scenarios. In many cases some missing transverse momentum is expected, e.g. due to neutrinos from the SM gauge bosons decays which arise in a cascade decays of vector like quarks, for example. The analyses measuring SM cross sections typically also require rather small missing transverse momentum.

Most searches contain multiple signal regions, e.g. the stop pair production search with one isolated lepton, jets and missing transverse momentum [69] has 27 signal regions. It is, therefore, sensitive to a large class of mass hierarchies between the stop, chargino NLSP and neutralino LSP. Counting all signal regions of the implemented searches, 190 signal regions are employed just for the ATLAS searches at 8 TeV. We would like to caution users that different signal regions across different analyses are in many cases not statistically independent. The correlations can be particularly strong for the signal regions with similar final states and kinematic cuts. Any statistical combination should take this into account.

²⁰Despite the description, this analysis allows for events with up to three hard jets in the final state and hence is also sensitive to our expected multijet signature.

ATLAS 8 TeV searches

Name	Search	\sqrt{s} [TeV]	\mathcal{L} [fb ⁻¹]	N_{SR}	Ref.	Cite
atlas_1308_1841	New phenomena in final states with large jet multiplicities and \cancel{E}_T	8	20.3	19	[62]	[63]
atlas_1308_2631	Third-generation squark pair production in final states with \cancel{E}_T and two b -jets	8	20.1	6	[64]	
atlas_1402_7029	Production of charginos and neutralinos in events with three leptons and \cancel{E}_T	8	20.3	24	[65]	
atlas_1403_4853	Top-squark pair production in final states with two leptons	8	20.3	12	[66]	
atlas_1403_5222	Top squark pair production in events with a Z boson, b -jets and \cancel{E}_T	8	20.3	5	[67]	
atlas_1404_2500	Supersymmetry in final states with jets and two same-sign leptons or three leptons	8	20.3	5	[68]	
atlas_1405_7875	Squarks and gluinos in final states with jets and \cancel{E}_T	8	20.3	15	[58]	[63]
atlas_1407_0583	Top squark pair production in final states with one isolated lepton, jets, and \cancel{E}_T	8	20.3	27	[69]	
atlas_1407_0608	Pair-produced third-generation squarks decaying via charm quarks or in compressed supersymmetric scenarios	8	20.3	3	[70]	
atlas_1411_1559	Monophoton search with one energetic photon and large \cancel{E}_T	8	20.3	1	[71]	
atlas_1501_07110	Search for direct pair production of a chargino and a neutralino decaying to the 125 GeV Higgs boson	8	20.3	12	[72]	
atlas_1502_01518	New phenomena in final states with an energetic jet and large \cancel{E}_T	8	20.3	9	[60]	
atlas_1503_03290	Supersymmetry in events containing a same-flavour opposite-sign dilepton pair, jets, and large \cancel{E}_T	8	20.3	1	[73]	[63]
atlas_1506_08616	Pair production of third-generation squarks	8	20.3	11	[74]	
atlas_conf_2012_104	Supersymmetry in final states with jets, \cancel{E}_T and one isolated lepton	8	5.8	2	[75]	
atlas_conf_2012_147	New phenomena in monojet plus \cancel{E}_T final states	8	10	4	[76]	
atlas_conf_2013_024	Production of the top squark in the all-hadronic $t\bar{t}$ and \cancel{E}_T final state	8	20.5	3	[77]	
atlas_conf_2013_049	Direct-slepton and direct-chargino production in final states with two opposite-sign leptons, \cancel{E}_T and no jets	8	20.3	9	[78]	
atlas_conf_2013_061	Strong production of supersymmetric particles in final states with \cancel{E}_T and at least three b -jets	8	20.1	9	[79]	
atlas_conf_2013_089	Strongly produced supersymmetric particles in decays with two leptons	8	20.3	12	[80]	
atlas_conf_2015_004	Invisibly decaying Higgs boson produced via vector boson fusion	8	20.3	1	[81]	

Table 3: List of 8 TeV ATLAS analyses which are available in the public alpha version of **CheckMATE** and which have been validated against published experimental results. The “Cite” column refers to an original paper by external authors who implemented a search and should be cited along with **CheckMATE**.

CMS 8 TeV searches

Name	Search	\sqrt{s} [TeV]	\mathcal{L} [fb ⁻¹]	N_{SR}	Ref.	Cite
cms_1303.2985	Supersymmetry in hadronic final states with missing transverse energy using the variables α_T and b -quark multiplicity	8	11.7	59	[61]	
cms_1408.3583	Dark matter, extra dimensions, and unparticles in monojet events	8	19.7	7	[82]	
cms_1502.06031	Physics beyond the Standard Model in events with two Leptons, jets, and \cancel{E}_T	8	19.4	6	[83]	[63]
cms_1504.03198	Production of dark matter in association with top-quark pairs in the single-lepton final state	8	19.7	1	[84]	[85]
cms_sus_13_016	New physics in events with same-sign dileptons and jets	8	19.5	1	[86]	

Table 4: List of 8 TeV CMS analyses which are available in the public alpha version of **CheckMATE** and which have been validated against published experimental results. The “Cite” column refers to an original paper by external authors who implemented a search.

ATLAS 13 TeV searches

Name	Search	\sqrt{s} [TeV]	\mathcal{L} [fb ⁻¹]	N_{SR}	Ref.	Cite
atlas_1602.09058	Supersymmetry in final states with jets and 2 same sign leptons or 3 leptons	13	3.2	4	[87]	
atlas_1604.01306	Search for new phenomena in events with a photon and missing transverse momentum	13	3.2	1	[88]	
atlas_1604.07773	Search for new phenomena in monojet events	13	3.2	13	[89]	
atlas_1605.03814	Squarks and gluinos in final states with 2 - 6 jets + \cancel{E}_T	13	3.2	7	[90]	
atlas_1605.04285	Gluino search in final states with an isolated lepton + jets + \cancel{E}_T	13	3.2	7	[91]	
atlas_1605.09318	Gluino pair productions in final states with b jets and \cancel{E}_T	13	3.2	3	[92]	
atlas_1606.03903	Stop search in final states with 1 lepton, jets and \cancel{E}_T	13	3.2	3	[93]	
atlas_conf_2015.082	Supersymmetry search in final states with leptonic Z + jets + \cancel{E}_T	13	3.2	1	[94]	
atlas_conf_2016.013	Vector like top quark production and 4 top quark production in lepton plus jets final state	13	3.2	10	[95]	
atlas_conf_2016.050	Search for top squarks in final states with one isolated lepton, jets, and \cancel{E}_T	13	13.2	5	[96]	
atlas_conf_2016.076	Search for direct top squark pair production and dark matter production in final states with two leptons	13	13.3	6	[97]	

Table 5: List of 13 TeV ATLAS analyses which are available in the public alpha version of **CheckMATE** and which have been validated against published experimental results.

CMS 13 TeV searches

Name	Search	\sqrt{s} [TeV]	\mathcal{L} [fb ⁻¹]	N_{SR}	Ref.	Cite
cms_pas_sus_15_011	Search for new physics in final states with two opposite-sign same-flavor leptons, jets and	13	2.2	47	[98]	

Table 6: List of 13 TeV CMS analyses which are available in the public alpha version of **CheckMATE** and which have been validated against published experimental results.

ATLAS 14 TeV high luminosity studies

Name	Search	\sqrt{s} [TeV]	\mathcal{L} [fb ⁻¹]	N_{SR}	Ref.	Cite
atlas_phys_2014_010_sq_h1	Gluino and squark production in final states with large jet multiplicities and \cancel{E}_T and with no leptons	14	3000	10	[99]	
atlas_phys_2014_010_300	Gluino and squark production in final states with large jet multiplicities and \cancel{E}_T and with no leptons	14	300	10	[99]	
atlas_phys_2014_010_h1_31	Direct production of charginos and neutralinos in final states with three leptons and \cancel{E}_T	14	3000	1	[99]	
atlas_phys_2014_010_sbotttom	Direct production of sbottom pairs in final states with $2b$ jets and \cancel{E}_T	14	3000	6	[99]	
atlas_phys_pub_013_011	Top squark pair production in final states with (b) jets, 0-1 lepton and \cancel{E}_T	14	3000	4	[100]	

Table 7: List of official ATLAS 14 TeV high luminosity analyses which are available in the public alpha version of **CheckMATE** and which have been validated against ATLAS MC results.

All analyses listed in the above tables are fully validated against published cut-flows, distributions and/or exclusion limit plots from both experimental collaborations. The validation notes can be found on the official **CheckMATE** webpage. The analyses are grouped into ATLAS and CMS searches at the center of mass energies of 8 and 13 TeV which are listed in Tables 3, 4, 5 and 6. It is clear from the tables that ATLAS outnumbers CMS in the number of implemented analyses. This is due to the fact that the efficiencies of the final state particles have only been optimized and fully validated for the ATLAS detector. However, some CMS searches are also included, especially when there is no ATLAS equivalent or are of particular interest for our own phenomenological studies. The number of signal regions and the total integrated luminosity are given for each analysis in the corresponding tables. More details of the implemented searches can be found in the respective references provided in the tables.

In addition to the fully validated analyses, **CheckMATE** also includes some analyses which have not been completely validated. This can be, for example, due to insufficient information from the collaborations. Therefore some of these analyses are only partially validated and the full list can be found on the **CheckMATE** web page. If available, partial validation notes may also be provided and these often contain details regarding the outstanding issues that are still to be solved. Obviously, some caution is necessary when using these analyses for physics studies, especially if the particular signal regions of interest have not been completely tested. Altogether there are currently about 60 analyses available in **CheckMATE** and the list is expanding rapidly.

Current searches already push the exclusion limits of gluinos and first generation squarks well beyond the TeV scale. In the high luminosity phase, the limits will significantly improve and are of general interest. Therefore official ATLAS SUSY high-luminosity studies at a centre-of-mass energy of 14 TeV for a total integrated luminosity of 300 and 3000 fb⁻¹ have now been included. Here, the high luminosity studies cover squark and gluino pair production, stop and sbottom pair production as well as chargino and neutralino production which are summarised in Table 7.

6. Performance Studies

FRITZ allows for a significant gain in performance by bypassing the generation of **HepMC** or **STDHEP** event files, as well as not storing the detector level objects in a **ROOT** file. This new module interfaces **Pythia**, **Delphes** and the **AnalysisHandler** without the necessity to write and read information on the hard disk as described in detail in section 2. Here, we compare the performance between **CheckMATE** 1 and **CheckMATE** 2. In both frameworks, MC events are generated with **Pythia** 8 and the events are stored in a **HepMC** file for the **CheckMATE** 1 setup whereas for the **CheckMATE** 2 case the MC events are directly passed to the **Delphes** module with **FRITZ**. The computations are performed on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60 GHz with 32 GB RAM. The performance has been quantified by generating 10000 gluino pairs at the LHC at the centre-of-mass energy of 8 TeV with underlying event, initial and final state radiation and hadronisation switched on. The current version **Pythia** 8.2.1.9 was employed for this purpose. The events were passed to **HepMC** 2.0.6.09 and the resulting event file had a size of 2.3 GB. The truth level MC event generation and the write operation to hard disk took 255 seconds in total. The **HepMC** file was then passed to **CheckMATE** 1.2.2 and tested against a single inclusive supersymmetry search [58] and it took 85 seconds for **CheckMATE** 1 to process the **HepMC** file. Thus the total processing time was 340 seconds. This time was compared to the computing time of **CheckMATE** 2.0.0. The same process with the same sparticle spectrum and the identical **Pythia** settings were employed. After generating 10000 events on the fly, the total computational time was 245 seconds which clearly shows the improved performance.

For the next comparison, several instances were run at the same time. Here, it becomes clear that **CheckMATE** 2 has a big advantage over **CheckMATE** 1 since simultaneous read and write operations significantly affect the performance. Ten **Pythia** instances linked with **CheckMATE** 1 were simultaneously run. 20000 events were generated with the same settings as before. The event generation with **Pythia** took about 564 seconds. The truth level events were passed to **CheckMATE** 1 and writing the root files for the detector level objects took 1237 seconds on average. Contrary to **CheckMATE** 2 the root file must have been created before the analysis step could have been performed. The total processing time was 1802 seconds. We repeated the test with **CheckMATE** 2 and the average running time was 544 seconds. The **CheckMATE** 2 run is

much faster since the reconstructed detector level objects are stored as ROOT objects which are immediately processed by the analysis module. It is evident that the bottleneck are the simultaneous write operations of ROOT files and this example clearly demonstrates the performance gain of **CheckMATE 2**.

7. Analysis Manager

Along with the main **CheckMATE** program, several improvements have also been made to the **AnalysisManager**. Since these are only minor updates to the already existing program we refer the reader to original manual [27] and here we only list the changes.

7.1. Prototyping New Analyses

The most significant addition to the **AnalysisManager** is the improved support for analysis prototyping i.e. developing new LHC analyses. Analyses can now be added (using the usual interactive procedure) without accompanying signal region data (numbers of observed/expected events etc.) to allow a user to first simulate various SM backgrounds. The various background contributions can then be added for each defined signal region and for the development of new analyses we assume the expected background and observed data are equal.

After all the Standard Model contributions have been calculated, the **AnalysisManager** can then be rerun with a newly added option to edit the analysis information. Here a user should enter the total Standard Model background for all signal regions defined in the analysis. At this step the **AnalysisManager** will internally calculate S_{95} limits so that new physics models can be quickly tested.²¹

7.2. Kinematical Variables

To enable a quick and easy implementation of analyses, **CheckMATE** contains a large library of kinematical variables that are often used by the LHC collaborations. Firstly, **CheckMATE** is interfaced with the **Delphes** and **ROOT** libraries and thus a large number of kinematical variables such as transverse momentum, energy, pseudorapidity, boosts, etc. are immediately available. A list of **Delphes** objects and their methods can be found at <https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/RootTreeDescription>. In addition, the **TLorentzVector** class from **ROOT** is included; see <https://root.cern.ch/root/html/TLorentzVector.html> for more details.

CheckMATE also contains a number of LHC mass-reconstruction variables which are directly implemented or included from other libraries, e.g. implementation of M_{T2} and derivatives [10, 11] and **MctLib** [12, 13]. A full list of these kinematical variables is given in Table 8 and further details can be found in our **doxygen** documentation <http://checkmate.hepforge.org/documentation/index.html>. See Ref. [101] for a review of the kinematical variables proposed for mass reconstruction and BSM searches at the LHC.

8. Summary

We have introduced the second version of the program **CheckMATE** which greatly improves the ease and speed with which models can be tested against the latest LHC data compared to its predecessor. The major improvement in this release is the integration of Monte Carlo event generation that allows a user to go directly from a model defined in the UFO format to the LHC results. In addition, this integration significantly reduces the CPU load required to investigate models.

Further improvements are the inclusion of over 50 analyses which now cover the vast majority of LHC searches that include missing energy. Moreover, high luminosity studies at 14 TeV are also included for the first time and these allow a user to understand the ultimate LHC reach for their model.

²¹By default the **AnalysisManager** also now internally calculates all S_{95} values rather than using those quoted by the experiments. The internally calculated values only show small differences compared to the numbers obtained the experiments. This approach ensures that the **CheckMATE** results are statistically consistent across all included analyses.

Kinematical variables

Name	Description/Example application	Ref.
M_T	Transverse Mass; reconstruction of W mass in $W \rightarrow \ell\nu$ decay	[102–106]
M_{T2}	Stransverse Mass; generalisation of M_T to events with more than one invisible particle	[8–10]
$M_{T2}^{b\ell}$	Asymmetric Stransverse Mass; suppression of SM top background	[10, 11]
M_{T2}^W	Asymmetric Stransverse Mass including W mass condition; suppression of SM top background	[10, 11]
M_{CT}	Cotransverse Mass; invariant under contra-linear transverse boosts	[12]
M_{CT} corrected	Cotransverse Mass corrected; takes initial state radiation into account	[13]
$M_{CT\perp}$ and $M_{CT\parallel}$	Decomposed Cotransverse Mass	[14]
α_T	Suppression of fake \cancel{E}_T in QCD events	[107, 108]
Razor	<i>Mega</i> -dijet kinematic variable without relying on \cancel{E}_T	[109–111]
‘Super’-Razor	Improved Razor that more accurately determines production and centre-of-mass frames	[16]
Topness	Suppression of SM top background	[15]
Aplanarity	Suppression of QCD events	[112]

Table 8: List of kinematical observables which are available in the public alpha version of **CheckMATE**.

For users who wish to include their own analyses or develop new LHC searches, the **AnalysisManager** also has a number of improvements to aid this process. As an example, backgrounds can be far more easily included and then used to test the reach of an analysis. In addition, the library of kinematical variables has grown significantly.

We should emphasise that the release of **CheckMATE 2** is simply a snapshot of a continuously evolving program. New analyses will regularly be included in updated versions available on the website. Besides, many new developments are also planned for the **CheckMATE** including a fast parameter scanning technique that does not require Monte Carlo events, automatic merging of matrix elements containing different jet multiplicities and the inclusion of systematic correlations between signal regions to allow proper combinations of many different analyses.

Acknowledgements

We would like to thank all of the following people for their contribution to **CheckMATE**.

- Especially Liangliang Shang for contributing analyses, performing validation studies and finding many bugs in **CheckMATE**.
- Junjie Cao, Jin Min Yang, Peiwen Wu, Jinmin Yang and Yang Zhang for contributing analyses to the **CheckMATE** database.
- Swasti Beswal, Anke Biekötter, Tim Keller and Jan Schütte-Engel for contributing analyses to the **CheckMATE** database.
- Sebastian Belkner for improving the statistical tools within **CheckMATE**.
- Florian Jetter for providing an updated muon resolution tuning.
- Daniel Antrim, Philip Bechtle, Jamie Boyd, Sascha Caron, Geraldine Conti, Carolina Deluca, Klaus Desch, Monica D’Onofrio, Till Eifert, Frank Filthaut, Eva Halkiadakis, Nicolai Hartman, Andreas Hoecker, Emma Kuwertz, Tommaso Lari, Zachary Marshall, Antoine Marzin, Federico Meloni, Alaettin Serhan Mete, Marija Vranjes Milosavljevic, Maurizio Pierini, Tina Potter, George Redlinger, Iacopo

Vivarelli, Steven Worm, Frank Wuerthwein and Takashi Yamanaka for help with interpreting the experimental analyses.

The work has been supported by the BMBF grant 00160200. ND acknowledges partial support of the OCEVU Labex (ANR-11-LABX-0060), the A*MIDEX project (ANR-11-IDEX-0001-02) funded by the French Government programme “Investissements d’Avenir” and the German Research Foundation (DFG) through the Forschergruppe *New Physics at the Large Hadron Collider* (FOR 2239). The work of JSK was supported by IBS under the project code, IBS-R018-D1 and was partially supported by the MINECO, Spain, under contract FPA2013-44773-P; Consolider-Ingenio CPAN CSD2007-00042 and the Spanish MINECO Centro de excelencia Severo Ochoa Program under grant SEV-2012-0249. KR was supported by the National Science Centre (Poland) under Grant 2015/19/D/ST2/03136 and the Collaborative Research Center SFB676 of the DFG, “Particles, Strings, and the Early Universe”.

A. Installation Instructions

CheckMATE uses a number of external programs and libraries.²² Below we split these into two categories, those that are always required and those that can be optionally installed to extend the functionality of the program. We also note that a step by step interactive online version is available at,

<http://checkmate.hepforge.org/tutorial/ver2/start.php>

This tutorial is particularly useful if some of the below steps should be skipped, either because some of the programs have already been installed on the system or if some optional parts are not required. Also, the online tutorial will be continuously updated if the below installation routines especially of the required additional libraries change.

A.1. Required Packages

CheckMATE requires **Python 2.7.X** where $X > 3$ (note that at the current time, **Python 3** is NOT supported), the data analysis package **ROOT** (v5.34.36 or later) [113] and the detector simulation **Delphes** (v3.3.3 or later) [1]. If any of these packages are already installed, the respective sections of the tutorial can be skipped. The physics specific programs can be downloaded from the relevant project websites,

<https://root.cern.ch>

<https://cp3.irmp.ucl.ac.be/projects/delphes>

We begin with the installation²³ of **ROOT** and we recommend that users do not install a binary version but rather compile the package from source into a specific installation directory. Here and in the following, we use [...] to denote the verbatim output created by the respective commands which we enter. These strongly depend on the system setup which is used. In the following we denote the installation directory by **\$ROOT**,

```
Terminal
$ROOT: mkdir build
$ROOT: mkdir build/etc
$ROOT: ./configure --prefix=$ROOT/build --disable-fftw3 --enable-minuit2 --etcdir=$ROOT/build/etc
[...]
$ROOT: make -j4
[...]
```

²²This tutorial has been tested on a Linux machine running under Ubuntu 16.04. The same source files can be used for other operating systems, however some flags might change or some additional system libraries might be required. We refer to the documentation pages of the respective tools and the **CheckMATE** website if problems of that kind occur.

²³Note that the `-j4` flag which we use here improves the compilation speed due to parallelisation into four independent processes. The number can be changed depending on the number of accessible cores on the computer.

```
$ROOT: make install
[...]
```

In order to build the **Delphes** detector simulation framework in the installation directory denoted by **\$DELPHES**, we have to load the above compiled ROOT libraries

```
Terminal
$DELPHES: source $ROOT/build/bin/thisroot.sh
$DELPHES: make -j4
[...]
```

A promising sign of a successful installation is the presence of the file **libDelphes.so** within the **Delphes** directory.

A.2. Optional Packages

CheckMATE 2 now includes various options to generate events with **Pythia 8** [7] and/or **MadGraph** [6]. In addition, if the user wishes to have the possibility to store the generated events in the **HepMC** format [114], the corresponding library needs to be installed first. These programs can be downloaded from the relevant project websites,

<http://hepmc.web.cern.ch/hepmc>
<http://home.thep.lu.se/~torbjorn/Pythia.html>
<https://launchpad.net/mg5amcnlo>

We start with the installation of the optional **HepMC** library which is only required if the user wishes to save generated events in this format. Here the installation directory is denoted by **\$HEPMC**,

```
Terminal
$HEPMC: ./bootstrap
[...]
```

```
$HEPMC: ./configure --with-momentum=GEV --with-length=MM --prefix=$HEPMC/build
[...]
```

```
$HEPMC: make -j4
[...]
```

```
$HEPMC: make install
[...]
```

If the installation finished successfully the **build** directory should contain the required libraries and header files which are needed by **Pythia 8**,

```
Terminal
$HEPMC: ls build
include lib share
```

We can continue with the compilation and installation of the **Pythia 8** event generator into the directory denoted by **\$PYTHIA8**. The command **--with-hepmc2=\$HEPMC/build** can be optionally removed if the **HepMC** library was not installed,

```
Terminal
$PYTHIA8: ./configure --with-hepmc2=$HEPMC/build --prefix=$PYTHIA8/build
[...]
```

```
$PYTHIA8: make -j4
[...]
```

```
$PYTHIA8: make install
[...]
```

Again, a successful installation procedure should have filled the **build** directory with the necessary library files

```

Terminal
$PYTHIA8: ls build
bin include lib share

```

The final optional program **CheckMATE** can link to is **MadGraph** and this only requires downloading and unzipping into the directory we denote by **\$MADGRAPH**.

A.3. Installing CheckMATE

With all required libraries being ready, we can finally compile the **CheckMATE** framework into the directory denoted by **\$CMMAIN**. In the following, any of the optional **--with** commands can be omitted if the relevant program has not been installed (note that **Delphes** and **Root** are required),

```

Terminal
$CMMAIN: ./configure --with-rootsys=$ROOT/build --with-delphes=$DELPHES --with-hepmc=$HEPMC/build \
--with-pythia=$PYTHIA8/build --with-madgraph=$MADGRAPH
[...]
make -j4
[...]
make install

```

Let us finish this tutorial with a simple test run, using an example spectrum file which is provided with the **CheckMATE** package. It corresponds to a CMSSM scenario with $\tan\beta = 10$, $m_0 = 100$ GeV, $m_{1/2} = 250$ GeV, $A_0 = -100$ GeV and positive μ . This results in a spectrum with all SUSY particles having mass in the range 100–600 GeV.

```

Terminal
$CMMAIN/bin: ./CheckMATE -pyp 'p p > go go' -maxev 100 \
-slha example_run_cards/auxiliary/testspectrum.slha
[...]
Is this correct? (y/n) y
[...]
Evaluating Results
Test: Calculation of r = signal/(95%CL limit on signal)
Result: Excluded
Result for r: 1.15744708205
Analysis: atlas_1405_7875
SR: SR03_a.4jl-

```

Such a SUSY scenario is so constrained by existing LHC searches that even a sample of only 100 Monte Carlo events is sufficient to exclude it within seconds.

B. Statistical Analysis in CheckMATE

A standard statistical problem which has to be solved in cut-based collider analyses is the following: What is the p -value of observing N events in a certain bin if the Standard Model (= alternative hypothesis) predicts $B \pm \Delta B$ events and the new physics model (= null hypothesis) predicts $S \pm \Delta S$ events in addition to B ? In this section we explain the exact procedure **CheckMATE** uses, based on the CL_s prescription paired with a likelihood ratio discriminator [5]. For a more detailed description, we refer to Ref. [115].

B.1. 1-bin Likelihood and Test Statistics

If S and B were known with infinite precision, the likelihood of observing N events in a bin where $S + B$ are theoretically expected would be given by the Poisson distribution

$$\mathcal{L}(N|S) = \text{Poiss}(N|S+B) \equiv \frac{(S+B)^N}{N!} e^{-(S+B)}. \quad (2)$$

In reality, we do not know the background for certain but repeated evaluations would result in different values $B_{\text{unc.}}$. This parameter is distributed according to a probability density function $P(B_{\text{unc.}}|B, \Delta_B)$ with

fixed B , Δ_B . Even though in principle there can be arbitrary many independent $(\Delta B)_i$ from different error sources, **CheckMATE** only considers the combined background error ΔB , cf. Section 4.3. The algorithm described below can however be straightforwardly extended.

It is a common practice to redefine $B_{\text{unc.}} = B_{\text{unc.}}(\theta)$ in terms of a dimensionless nuisance parameter θ . It is then θ which is distributed according to a density function $P(\theta|\tilde{\theta})$ with $\tilde{\theta}$ being the *a priori* most probable value. At the beginning, this value is trivially given via $B_{\text{unc.}}(\tilde{\theta}) = B$, however along the calculation of CL_S, the value of $\tilde{\theta}$ will change as described below.

In **CheckMATE**, we always assume $B_{\text{unc.}}(\theta) = B \exp(\theta \Delta B / B)$ and θ to be Gaussianly distributed according to $P(\theta|\tilde{\theta}) \propto \exp(-(\theta - \tilde{\theta})^2 / 2)$. With this choice of parameters, the *a priori* value for θ is 0. For small $\Delta B / B$, this *lognormal distribution* will lead to a Gaussianly distributed $B_{\text{unc.}}$ with mean B and standard deviation ΔB . However, for very large uncertainties it prevents $B(\theta)$ from turning to unphysical negative values.

Following the same approach for $S_{\text{unc.}}(\theta)$ and assigning independent uncertainties to signal and background we get the following extended likelihood:²⁴

$$\mathcal{L}(N, \tilde{\theta}_S, \tilde{\theta}_B | \mu, \theta_B, \theta_S) \equiv \left(\frac{1}{N!} \left[\lambda(\mu, \theta_B, \theta_S) \right]^N e^{-\lambda(\mu, \theta_B, \theta_S)} \right) \cdot \left(e^{-(\tilde{\theta}_B - \theta_B)^2 / 2} \right) \cdot \left(e^{-(\tilde{\theta}_S - \theta_S)^2 / 2} \right), \quad (3)$$

$$\lambda(\mu, \theta_B, \theta_S) \equiv \mu S e^{\left(\frac{\Delta_S}{S} \theta_S \right)} + B e^{\left(\frac{\Delta_B}{B} \theta_B \right)}. \quad (4)$$

Note that we have introduced the signal strength modifier μ , which will prove more convenient in distinguishing signal and background hypotheses for varying $S_{\text{unc.}}(\theta_S)$.

There are different approaches to incorporate the unknown nuisance parameters θ_B, θ_S into the test statistics. **CheckMATE** uses the *Profile Likelihood Ratio* defined as

$$q_\mu(N, \tilde{\theta}_S, \tilde{\theta}_B) \equiv -2 \log \left(\frac{\mathcal{L}(N, \tilde{\theta}_S, \tilde{\theta}_B | \mu, \hat{\theta}_S^\mu, \hat{\theta}_B^\mu)}{\mathcal{L}(N, \tilde{\theta}_S, \tilde{\theta}_B | \hat{\mu}, \hat{\theta}_S, \hat{\theta}_B)} \right). \quad (5)$$

Here, $\hat{\mu} \in [0, \mu]$ ²⁵, $\hat{\theta}_S$ and $\hat{\theta}_B$ is the combination of all three parameters which globally maximises \mathcal{L} , whereas $\hat{\theta}_S^\mu, \hat{\theta}_B^\mu$ are the values which maximise $\mathcal{L}(\mu)$ for fixed μ . $q_\mu(N, \tilde{\theta}_S, \tilde{\theta}_B)$ becomes larger for less compatibility of observation and null hypothesis. According to Wilks' theorem [116], the maximum likelihood ratio approaches a χ^2 -distribution for large event rates.

Note that even for our rather simple statistical setup, the numerator cannot be evaluated analytically. As such, **CheckMATE** uses the numerical `scipy.optimize.root` routine to find the roots of the first derivatives in order to evaluate the test statistics.

B.2. Confidence Levels and p-values

With the test statistics of Eq. (5), we can determine the *p*-value of the signal hypothesis $S + B$ after observing N as follows. If we repeated the experiment infinitely many times we would expect to observe different values N' each time such an experiment is performed due to the statistical nature of the underlying physics. Also, our determination of the distributions for $B_{\text{unc.}}(\theta_B), S_{\text{unc.}}(\theta_S)$ would have resulted in different values for the expectation values θ_B, θ_S which we *a priori* assumed to be 0. The values which we would expect depend on the underlying hypothesis we assume.

If the signal hypothesis $\mu = 1$ was correct, according to the likelihood in Eq. (3) the most compatible values for the nuisance parameters θ_S, θ_B after the observation of N would be their best fit values $\hat{\theta}_S^{\mu=1}$ and $\hat{\theta}_B^{\mu=1}$. If we thus *a posteriori* assume that these were the true expected values of $\tilde{\theta}_S, \tilde{\theta}_B$ and we hypothetically redid the experiment we would expect a random value of $\tilde{\theta}_B$ according to a Gaussian distribution with

²⁴Note that we can safely ignore normalisation factors for the Gaussian distributions as they will not contribute to the likelihood ratio.

²⁵The lower limit $\hat{\mu} \geq 0$ leads to a one-sided limit on S while the upper limit $\hat{\mu} \leq \mu$ ensures that the test statistics is 0 if the global best fit would prefer an even larger signal than the one tested. In other words, we claim perfect compatibility of signal hypothesis and observation also if a larger signal would fit the observation better.

expectation value $\hat{\theta}_B^{\mu=1}$, analogously for $\tilde{\theta}_S$. Also, the number of observed events N' should be Poisson distributed with the expectation value $\lambda(\mu=1, \hat{\theta}_S^{\mu=1}, \hat{\theta}_B^{\mu=1})$.

For a p -value we are interested in the fraction of these hypothetical experiments which would perform in a test statistics at least as bad as the observed one. We call this value for the signal hypothesis CL_{S+B} and formulate it analytically as follows:

$$\text{CL}_{S+B} \equiv \sum_{N'=0}^{\infty} \int_{-\infty}^{\infty} d\tilde{\theta}'_S \int_{-\infty}^{\infty} d\tilde{\theta}'_B \Theta\left(q_{\mu=1}(N', \tilde{\theta}'_S, \tilde{\theta}'_B) - q_{\mu=1}(N, \tilde{\theta}_S, \tilde{\theta}_B)\right) \times \\ \text{Pois}(N'|\lambda(\mu=1, \hat{\theta}_S^{\mu=1}, \hat{\theta}_B^{\mu=1}) \cdot \text{Gauss}(\tilde{\theta}'_S|\hat{\theta}_S^{\mu=1}) \cdot \text{Gauss}(\tilde{\theta}'_B|\hat{\theta}_B^{\mu=1}), \quad (6)$$

where we define $\Theta(x) = 1$ for $x \geq 0$ and 0 else.

This approach however has a peculiar property: If, for example, N happens to be much smaller than $B - 2\Delta B$, which statistically can happen in a small fraction of experiments even if B describes Nature accurately, CL_{S+B} will always turn out to be small, regardless of S . Therefore, the above interpretation will always claim a tension with the signal hypothesis and could even conclude that a model with $S \ll \Delta B$ is excluded. One should be worried in this case as it intuitively sounds incorrect to conclude anything about a signal which is much smaller than the systematic uncertainty of the experiment.

A commonly used approach to avoid such a false exclusion is to, in addition to CL_{S+B} , determine the p -value for the observation to be compatible with the background-only hypothesis. For that purpose, we simply set $\mu = 0$ in our above explanation and thus evaluate

$$1 - \text{CL}_B = \sum_{N'=0}^{\infty} \int_{-\infty}^{\infty} d\tilde{\theta}'_S \int_{-\infty}^{\infty} d\tilde{\theta}'_B \Theta\left(q_{\mu=1}(N', \tilde{\theta}'_S, \tilde{\theta}'_B) - q_{\mu=1}(N, \tilde{\theta}_S, \tilde{\theta}_B)\right) \times \\ \text{Pois}(N'|\lambda(\mu=0, \hat{\theta}_S^{\mu=0}, \hat{\theta}_B^{\mu=0}) \cdot \text{Gauss}(\tilde{\theta}'_S|\hat{\theta}_S^{\mu=0}) \cdot \text{Gauss}(\tilde{\theta}'_B|\hat{\theta}_B^{\mu=0}). \quad (7)$$

Note that it is a common misconception to evaluate the test statistics of $1 - \text{CL}_B$ with $q_{\mu=0}$ instead of $q_{\mu=1}$. However, through the entire limit setting procedure we are testing the signal hypothesis and at this stage we are estimating what the result of this test would be if the background hypothesis was true. This is why we use $\mu = 0$ for the expectation values of N' , $\tilde{\theta}_S$ and $\tilde{\theta}_B$ but still need to evaluate the test statistics for the signal hypothesis $\mu = 1$.

In the CL_S prescription, the confidence in the signal hypothesis is calculated by the ratio

$$\text{CL}_S \equiv \frac{\text{CL}_{S+B}}{1 - \text{CL}_B} \quad (8)$$

and we interpret this value as the p -value for our signal. Consequently, we exclude a signal model if it produces a too small CL_S value. If an experiment shows a perfect agreement with the Standard Model prediction, $1 - \text{CL}_B$ equals 0.5 and therefore CL_S is very close to the true p -value CL_{S+B} . For experiments which are in tension with the background-only hypothesis, $1 - \text{CL}_B$ becomes smaller, CL_S increases and thus the limit weakens. CL_S therefore sets a conservative limit in the presence of under-fluctuations in the data.

CheckMATE evaluates the integrals in Eqs. (6) and (7) numerically by generating tuples of random numbers $N', \tilde{\theta}_B, \tilde{\theta}_S$ according to the respective Poisson or Gaussian distribution and counting the relative amount of tuples which yield a value of $q_{\mu=1}$ smaller than $q_{\mu=1}(N, 0, 0)$.

B.3. Model Independent Limits S95

In the main text, we explained how **CheckMATE** usually does not calculate CL_S in each run but normally calculates $r = (S - 1.64 \cdot \Delta S)/S95$ with the model independent upper 95% confidence limit $S95$. As the value for ΔS is not known in that case, it is set to 0 for the following evaluation. **CheckMATE** uses simplified versions of the above described formulae with all terms depending on θ_S removed and using $\lambda(\mu, \theta_B) \equiv \mu S + B \exp(\frac{\Delta B}{B} \theta_B)$.

For the *observed* limit, `S95_obs`, one simply needs to find the value of S which yields $\text{CL}_S = 0.05$. `CheckMATE` uses the Pegasus regula-falsi method for this purpose [117]. For the *expected* limit, `S95_exp`, `CheckMATE` determines the limit if we observed what we derived as the true background values from our observation, see Section B.2 above. That means we set $N = \lambda(\mu = 0, \hat{\theta}_B^{\mu=0})$ and the *a priori* $\tilde{\theta}_B$ to $\hat{\theta}_B^{\mu=0}$ and follow the same prescription as before.

B.4. Likelihood

As mentioned in Section 3, `CheckMATE` is capable of returning a combined likelihood summed over all bins. Here, it calculates the test statistics as in Eq. (5) and sums the result over all bins. For this purpose, we remove the restriction $\hat{\mu} \in [0, \mu]$ from Eq. (5) as we are not trying to perform a one-sided signal test. This calculation can be done for all bins and the sum of all likelihoods over all considered signal regions is returned by `CheckMATE`.

We note that a user should exercise care with the likelihood calculation since it can only be applied to orthogonal signal regions and some of the analyses included in `CheckMATE` do not fulfil this condition. If the user wishes to include such analyses, the likelihood sum should be calculated manually by selecting the result of the relevant orthogonal signal regions.

C. Tuning

In this appendix we provide details of the new `CheckMATE` tunings for lepton efficiencies and b -tagging. In the case of leptons we use recent ATLAS parametrisations updated during the 8 and 13 TeV runs. The new lepton energy resolutions have been implemented in the `Delphes` cards for 13 and 14 TeV analyses. The efficiency is evaluated internally by the respective `AnalysisHandlers`.

C.1. Electrons

Electron identification in the ATLAS detector is based on the multivariate analysis (MVA) that simultaneously evaluates several properties of the electron candidates. Three levels of identification are implemented in `CheckMATE` that correspond to the typical ATLAS operating points: loose, medium and tight. The efficiency depends on the transverse energy of the electron candidate, E_T , and to a lesser extent on the rapidity [118]. The current implementation in the `AnalysisHandler` takes into account only the former. The following functions are used for the 13 TeV setup:

$$\epsilon_{\text{id}}^{\text{loose}} = 0.976 - 0.0614 \cdot \exp\left(1 - \frac{E_T}{29.1}\right), \quad (9)$$

$$\epsilon_{\text{id}}^{\text{medium}} = 0.937 - 0.109 \cdot \exp\left(1 - \frac{E_T}{21.0}\right), \quad (10)$$

$$\epsilon_{\text{id}}^{\text{tight}} = 0.8885 - 0.138 \cdot \exp\left(1 - \frac{E_T}{27.5}\right), \quad (11)$$

where E_T is the transverse energy of the candidate in GeV. The parameters in the above parametrisation were obtained from the fit to the efficiency plots reported by ATLAS [118, 119]. Figure 2 shows the electron reconstruction and identification efficiency reported by ATLAS and obtained with `CheckMATE` for tight, medium and loose electrons as a function of electron candidate transverse energy.

For the parametrisation of electron energy resolution we follow Ref. [120]. The functions describing smearing of energy and momenta are the following:

$$\sigma(\text{GeV}) = 0.3 \oplus 0.10 \times \sqrt{E(\text{GeV})} \oplus 0.010 \times E(\text{GeV}) \quad \text{for } |\eta| < 1.4, \quad (12)$$

$$\sigma(\text{GeV}) = 0.3 \oplus 0.15 \times \sqrt{E(\text{GeV})} \oplus 0.015 \times E(\text{GeV}) \quad \text{for } 1.4 < |\eta| < 2.47. \quad (13)$$

Here, $a \oplus b \equiv \sqrt{a^2 + b^2}$. The functions are implemented in the `delphes_skimmed_ATLAS_13TeV.tcl` and `delphes_skimmed_ATLAS_14TeV.tcl` cards.

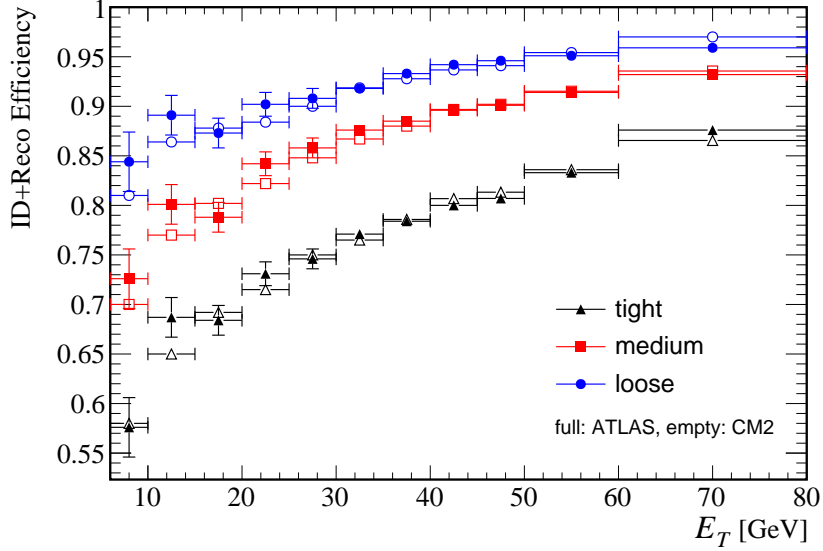


Figure 2: Comparison of the combined electron reconstruction and identification efficiency reported by ATLAS (full markers; see Figure 14 of Ref. [118]) and obtained with **CheckMATE** (empty markers) for tight (black triangle), medium (red square) and loose (blue circle) electrons as a function of electron candidate transverse energy and in the full pseudorapidity range, $|\eta| < 2.47$.

C.2. Muons

Muon reconstruction and identification efficiency shows generally an excellent performance across different rapidities and energy ranges [121]. In **CheckMATE** the efficiency is about 0.99 for loose muons and 0.97 for tight muons, except for muon candidates with $|\eta| < 0.1$ where it is set to 0.6. The dependence on p_T , unlike for the electrons, is negligible.

The combined muons correspond to muons observed in the inner detector (ID) and in the muon spectrometer (MS). The resolution formula combines the information from both systems and is given by [122]:

$$\sigma_{\text{ID}} = p_T \times \sqrt{a_1^2 + (a_2 \times p_T)^2}, \quad (14)$$

$$\sigma_{\text{MS}} = p_T \times \sqrt{\left(\frac{b_0}{p_T}\right)^2 + b_1^2 + (b_2 \times p_T)^2}, \quad (15)$$

$$\sigma_{\text{CB}} = \frac{\sigma_{\text{ID}} \times \sigma_{\text{MS}}}{\sqrt{\sigma_{\text{ID}}^2 + \sigma_{\text{MS}}^2}}, \quad (16)$$

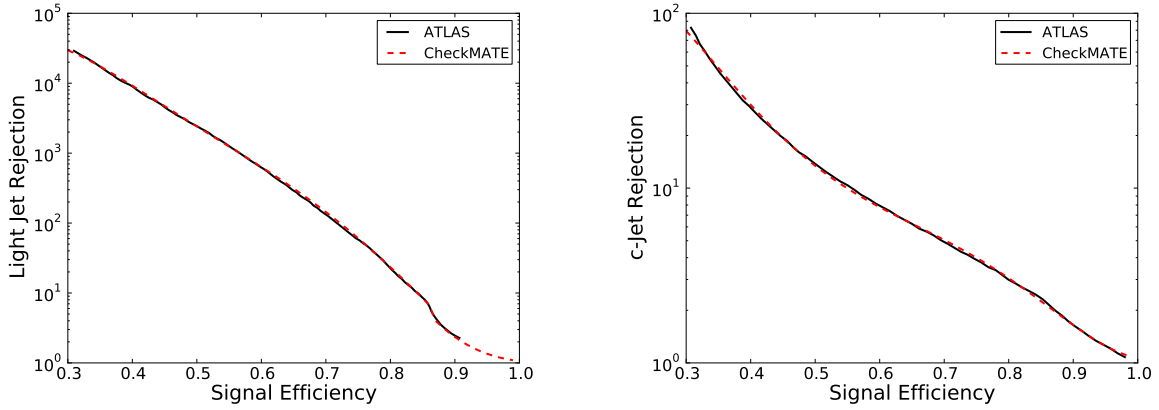
where p_T is the truth transverse momentum in GeV. The coefficients b_0 , b_1 , b_2 are specified in Table 9 and are the same for 13 and 14 TeV analyses. In the same table we also provide coefficients a_1 and a_2 for the 13 TeV setup. For the HL option the a_1 and a_2 coefficients are specified in 15 separate regions in rapidity taking into account planned upgrades to the inner detector. The full list can be found in Ref. [122].

C.3. B-Tagger

The quality of algorithms that try to filter jets containing b -quarks from others is determined by two main quantities. The signal efficiency describes the probability to assign a tag to a jet that actually contains a b -quark, whereas the background efficiency is a measure for the relative amount of jets that are tagged even though they do not have any bottom quark content. Since the background efficiency is usually small, it

	a_1	a_2	b_0	b_1	b_2
$ \eta < 1.05$	0.01607	0.000307	0.24	0.02676	0.00012
$ \eta > 1.05$	0.03000	0.000387	0.00	0.03880	0.00016

Table 9: The muon resolution coefficients a_1 , a_2 , b_0 , b_1 , b_2 from Eqs. (14) and (15) for the 13 TeV ATLAS setup.



(a) Rejection curve for jets containing light quarks only.

(b) Rejection curve for jets with charm content.

Figure 3: Receiver Operation Characteristic curves for a dependence of the background rejection for jets with different quark contents on a chosen signal efficiency working point of the b -tagger [123].

is common to use the inverse value, called rejection, for illustrative purposes. Also, one usually distinguishes between rejections against jets with charm content and other jets that only contain light quarks, as the first are harder to distinguish from the signal.

Since the rejection gets weaker with increasing signal efficiency, one has to find a balance between signal quantity and signal purity, which depends crucially on the details of the respective analysis. For this purpose, one uses the ROC (Receiver Operation Characteristic) curve that describes the relation between these two quantities.

C.3.1. 8 TeV

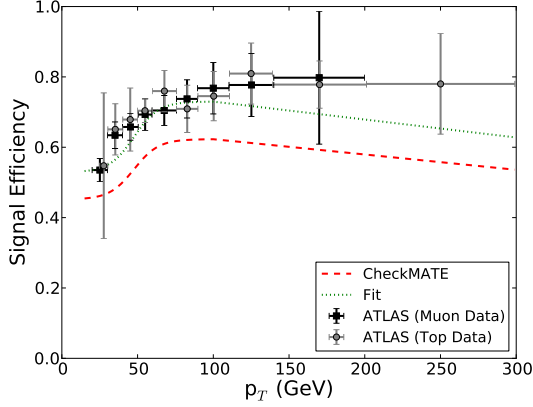
We show the ROC curves for light-jet and c -jet rejection separately in Figure 3 and internally parametrise these as follows:

$$\log_{10} [\bar{r}_{\text{light}}(\bar{\epsilon}_S)] = \begin{cases} 52.8 \cdot (\bar{\epsilon}_S - 4.045 \cdot \bar{\epsilon}_S^2 + 7.17 \cdot \bar{\epsilon}_S^3 - 6.14 \cdot \bar{\epsilon}_S^4 + 2.01 \cdot \bar{\epsilon}_S^5) & \text{if } \bar{\epsilon}_S < 0.87, \\ -75.4 \cdot (\bar{\epsilon}_S - 1.07)^3 & \text{if } \bar{\epsilon}_S \geq 0.87, \end{cases} \quad (17)$$

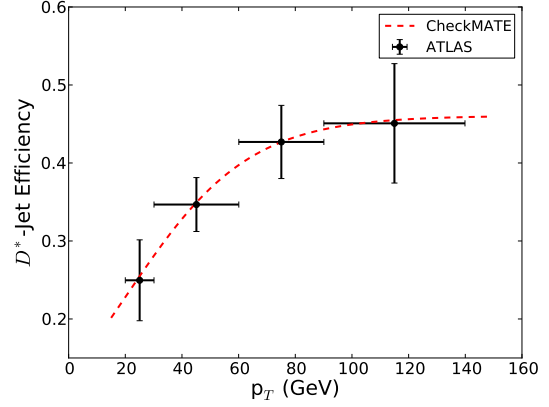
$$\log_{10} [\bar{r}_c(\bar{\epsilon}_S)] = 29.3 \cdot (\bar{\epsilon}_S - 4.572 \cdot \bar{\epsilon}_S^2 + 8.496 \cdot \bar{\epsilon}_S^3 - 7.253 \cdot \bar{\epsilon}_S^4 + 2.33 \cdot \bar{\epsilon}_S^5), \quad (18)$$

where $\bar{\epsilon}_S$ is signal efficiency and $\bar{r}_{\text{light}/c}$ background rejection for light and c -jets, respectively.

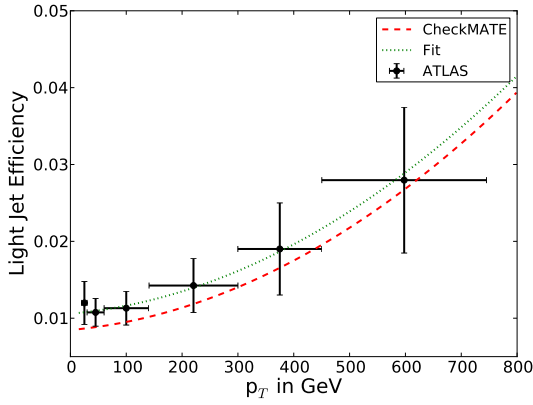
Given a particular working point on the ROC curve, i.e. a specific chosen signal efficiency, $\bar{\epsilon}_S$, and the corresponding background rejections, $\bar{r}_{\text{light}/c}$, the actual tagging probabilities depend on the transverse momentum of the considered object. These have been measured individually for signal, light-quark and D^*



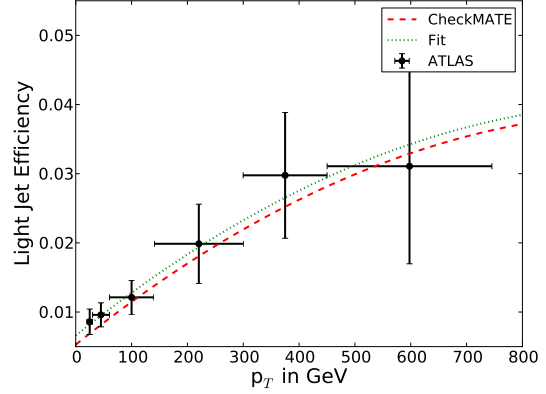
(a) Signal efficiencies for b -tagging, determined by combining the information from two different search channels for $\bar{\epsilon}_S = 0.7$ [123, 124].



(b) b -tagging efficiency of jets containing D^* mesons for $\bar{\epsilon}_S = 0.7$ [124]. The inclusive efficiency for c -jets is assumed to be 40% of this D^* efficiency.



(c) b -tagging efficiency of jets containing light quarks for $\bar{\epsilon}_S = 0.7$ and $|\eta| < 1.3$ [125].



(d) Same as (c) for $1.3 \leq |\eta| < 2.5$.

Figure 4: Dependence on the signal and light-jet / c -jet background efficiencies for b -tagging on the transverse momentum of the jet candidate.

meson²⁶ jets and we show the results in Figure 4.

For the signal efficiency, we use two different data sets as they have different sensitivities at low and high energies; see Figure 4a. In order to agree with the cutflows of various analyses that require b -tagging, a reduction in the overall normalisation by 15% has been applied. In addition, a significant decrease of the signal efficiency at large energies has been manually added in order to get a better agreement with experimental results. Furthermore, the light-quark jet rejection has been measured for two different η regions, which we adapt in our parametrisations. We also perform a reduction in the light-quark tagging rates (20%) in order to better agree with experimental cutflows.

²⁶The tagging probability for jets containing D^* mesons is roughly 2 times higher than for other c -mesons. Using the cutflows from various analyses we have tuned this parameter to 0.4 to be in agreement with the ATLAS results.

	$a_{i,0}$	$a_{i,1}$	$a_{i,2}$	$b_{i,1}$	$b_{i,2}$
light	$6.77 \cdot 10^{-6}$	$-9.86 \cdot 10^{-6}$	$2.79 \cdot 10^{-5}$	$-1.99 \cdot 10^2$	$2.50 \cdot 10^2$
charm	$-2.97 \cdot 10^{-3}$	$-2.15 \cdot 10^{-6}$	$1.29 \cdot 10^{-4}$	$-1.43 \cdot 10^2$	$1.12 \cdot 10^2$

Table 10: The coefficients for the light- and charm-jet efficiency as a function of η , Eq. (22).

Since the p_T dependent distributions are given for a particular working point $\bar{\epsilon}_S = 0.7$, we linearly rescale the functions to the given chosen signal efficiency $\bar{\epsilon}_S$, or the corresponding background efficiency given by the ROC curves (p_T in GeV):

$$\epsilon_S(p_T) = \frac{\bar{\epsilon}_S \cdot 0.85}{0.7} \left(0.552 + \frac{0.210}{1 + e^{-0.123 \cdot (p_T - 47.6)}} \right) \times \frac{0.7 + 0.05 \cdot e^{-\frac{p_T}{308}}}{0.75} \begin{cases} 1 & \text{if } p_T \leq 100, \\ 1 - 7 \times 10^{-4} \cdot (p_T - 100) & \text{if } p_T > 100, \end{cases} \quad (19)$$

$$\epsilon_{\text{light}}(p_T, \eta) = \frac{\bar{r}_{\text{light}}(0.7) \times 0.8}{\bar{r}_{\text{light}}(\bar{\epsilon}_S)} \begin{cases} 1.06 \times 10^{-2} + 6.47 \times 10^{-6} \cdot p_T^2 + 4.03 \times 10^{-8} \cdot p_T^4 & \text{if } |\eta| < 1.3, \\ 6.61 \times 10^{-3} + 6.49 \times 10^{-5} \cdot p_T^2 - 3.12 \times 10^{-8} \cdot p_T^4 & \text{if } 1.3 \leq |\eta| < 2.5, \end{cases} \quad (20)$$

$$\epsilon_c(p_T) = \frac{\bar{r}_c(0.7) \times 0.4}{\bar{r}_c(\bar{\epsilon}_S)} \frac{0.461}{1 + e^{-0.0464 \cdot (p_T - 20.4)}}. \quad (21)$$

C.3.2. 13 TeV

For the ATLAS analyses performed in Run-2, **CheckMATE** uses efficiencies fitted to the MV2c20 algorithm described in [126]. The dependence of the b -jet efficiency on p_T is kept the same as in Run-1, but the mistag rates differ from their Run-1 equivalents.

The light- and charm-jet efficiencies in **CheckMATE** depend on both η and p_T , with the following functional dependencies. The dependence on the pseudorapidity η is as follows:

$$\epsilon_i(\eta, \bar{\epsilon}_S) = (a_{i,0} + a_{i,1}\eta + a_{i,2}\eta^2) \cdot (1 + b_{i,1}\bar{\epsilon}_S + b_{i,2}\bar{\epsilon}_S^2) \cdot \frac{\bar{r}_i(0.7)}{\bar{r}_i(\bar{\epsilon}_S)}, \quad (22)$$

where i is either c or l for charm and light jets, respectively. The coefficients of these efficiency functions are shown in Table 10 and the functional dependence on the b -efficiency is included through the ROC curves \bar{r}_i , given as

$$\log_{10} [\bar{r}_{\text{light}}(\bar{\epsilon}_S)] = -21.9 \bar{\epsilon}_S^2 + 14.5 \bar{\epsilon}_S + 7.02, \quad (23)$$

$$\log_{10} [\bar{r}_c(\bar{\epsilon}_S)] = 7.39 \bar{\epsilon}_S^2 - 19.7 \bar{\epsilon}_S + 12.4. \quad (24)$$

The fitted functions given by Eq. (22) are shown in Figures 5b and 6b as solid lines, compared to the values from ATLAS, shown as dots.

The p_T dependence is given by a piecewise function, where each piece depends on the p_T of the jet and on the b -efficiency. The form of the pieces is

$$\epsilon_{i,\alpha}(p_T, \bar{\epsilon}_S) = [(a_{i,\alpha} + b_{i,\alpha}\bar{\epsilon}_S + c_{i,\alpha}\bar{\epsilon}_S^2) + (d_{i,\alpha} + e_{i,\alpha}\bar{\epsilon}_S + f_{i,\alpha}\bar{\epsilon}_S^2)p_T] \cdot \frac{\epsilon_{s,i}(\bar{\epsilon}_S)}{\epsilon_{s,i}(0.7)}, \quad (25)$$

where i is either c or l for charm and light jets, respectively, and α enumerates the pieces of the function. The coefficients for light and charm jets are given in Tables 11 and 12.

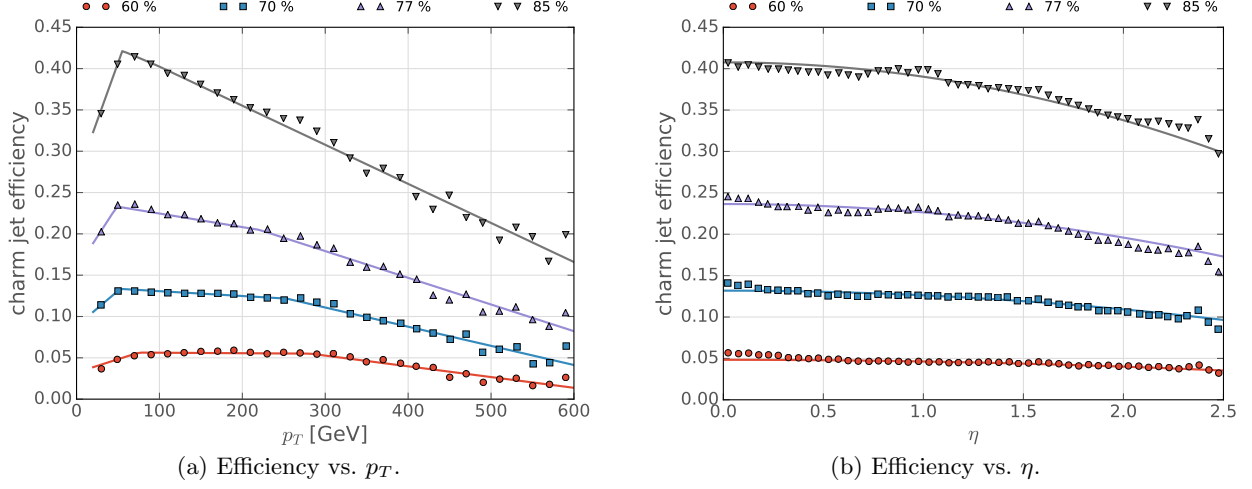


Figure 5: The charm-jet efficiency for the 13 TeV ATLAS b -tagger, for four different working points. The points show the expected efficiencies as determined by the ATLAS collaboration. The solid lines are the functions that **CheckMATE 2** uses to model the efficiencies.

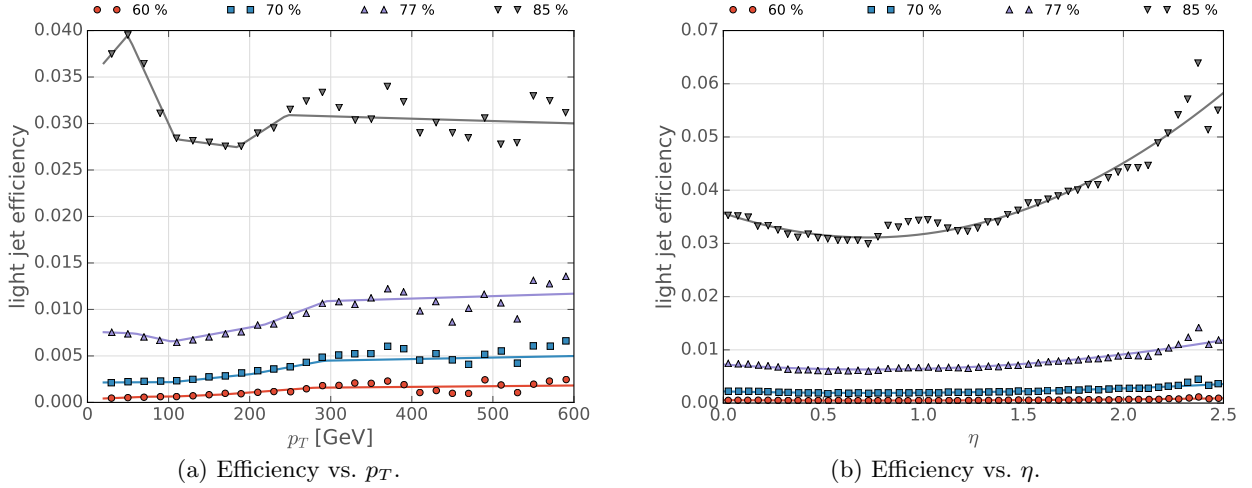


Figure 6: The light-jet efficiency for the 13 TeV ATLAS b -tagger, for four different working points. The points show the expected efficiencies as determined by the ATLAS collaboration. The solid lines are the functions that **CheckMATE 2** uses to model the efficiencies.

Outside of the range, $\bar{\epsilon}_S \in [0.6, 0.85]$, the parameters are frozen to either 60 % or 85 % and the scaling is exclusively given by the scale functions \bar{r}_i given by Eqs. (23) and (24). The fitted functions are shown in Figures 5a and 6a as solid lines, compared to the points they were fitted to. The efficiency functions of p_T and η are combined into a single function of p_T and η , and normalised using 12 million $t\bar{t}$ events,

$$\epsilon_{\text{light}}(p_T, \eta, \bar{\epsilon}_S) = \epsilon_{\text{light}}(p_T, \bar{\epsilon}_S) \cdot \epsilon_{\text{light}}(\eta, \bar{\epsilon}_S) \cdot 450 \frac{\bar{r}_{\text{light}}(\bar{\epsilon}_S)}{\bar{r}_{\text{light}}(0.7)} \left(1 + \frac{3(100\bar{\epsilon}_S - 60)}{1000} \right), \quad (26)$$

$$\epsilon_c(p_T, \eta, \bar{\epsilon}_S) = \epsilon_c(p_T, \bar{\epsilon}_S) \cdot \epsilon_c(\eta, \bar{\epsilon}_S) \cdot 7.5 \frac{\bar{r}_c(\bar{\epsilon}_S)}{\bar{r}_c(0.7)} \left(1 + \frac{100\bar{\epsilon}_S - 60}{130} \right). \quad (27)$$

These efficiencies were tested against the subset of the 13 TeV analyses implemented in **CheckMATE 2** that

α	1	2	3	4	5
$a_{\text{light},\alpha}$	-0.0152	$2.98 \cdot 10^{-3}$	$2.40 \cdot 10^{-3}$	$2.40 \cdot 10^{-3}$	-0.0153
$b_{\text{light},\alpha}$	0.0228	$-2.57 \cdot 10^{-3}$	$-4.31 \cdot 10^{-4}$	$-4.31 \cdot 10^{-3}$	0.0147
$c_{\text{light},\alpha}$	-0.00637	$2.70 \cdot 10^{-3}$	0	0	0
$d_{\text{light},\alpha}$	0.000384	$2.26 \cdot 10^{-5}$	$3.53 \cdot 10^{-5}$	$1.13 \cdot 10^{-4}$	$1.43 \cdot 10^{-5}$
$e_{\text{light},\alpha}$	-0.00103	$-1.64 \cdot 10^{-5}$	$-1.63 \cdot 10^{-5}$	$-1.82 \cdot 10^{-4}$	$-2.32 \cdot 10^{-5}$
$f_{\text{light},\alpha}$	0.000684	$-2.63 \cdot 10^{-5}$	$-3.04 \cdot 10^{-5}$	$6.20 \cdot 10^{-5}$	$0.74 \cdot 10^{-5}$

Table 11: Coefficients for the p_T -dependent light-jet efficiency function, Eq. (25).

α	1	2	3
$a_{c,\alpha}$	-0.288	0.576	1.51
$b_{c,\alpha}$	0.375	-0.941	-2.67
$c_{c,\alpha}$	-0.0329	0.513	1.31
$d_{c,\alpha}$	$1.02 \cdot 10^{-3}$	$2.13 \cdot 10^{-4}$	$-2.13 \cdot 10^{-3}$
$e_{c,\alpha}$	$-2.09 \cdot 10^{-4}$	$-3.52 \cdot 10^{-4}$	$4.43 \cdot 10^{-3}$
$f_{c,\alpha}$	$-5.00 \cdot 10^{-5}$	$-5.20 \cdot 10^{-5}$	$-2.45 \cdot 10^{-3}$

Table 12: Coefficients for the p_T -dependent charm-jet efficiency function, Eq. (25).

use b -jets. The validation for this class of analyses was performed using the b -tagger described here and the results were found to be in a good agreement with the published results.

References

- [1] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, M. Selvaggi, DELPHES 3, A modular framework for fast simulation of a generic collider experiment, JHEP 02 (2014) 057. [arXiv:1307.6346](#).
- [2] M. Cacciari, G. P. Salam, G. Soyez, FastJet User Manual, Eur. Phys. J. C72 (2012) 1896. [arXiv:1111.6097](#).
- [3] M. Cacciari, G. P. Salam, Dispelling the N^3 myth for the k_t jet-finder, Phys. Lett. B641 (2006) 57–61. [arXiv:hep-ph/0512210](#).
- [4] M. Cacciari, G. P. Salam, G. Soyez, The anti- k_t jet clustering algorithm, JHEP 0804 (2008) 063. [arXiv:0802.1189](#).
- [5] A. L. Read, Presentation of search results: The CL_S technique, J. Phys. G28 (2002) 2693–2704.
- [6] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, JHEP 07 (2014) 079. [arXiv:1405.0301](#).
- [7] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, P. Z. Skands, An Introduction to PYTHIA 8.2, Comput. Phys. Commun. 191 (2015) 159–177. [arXiv:1410.3012](#).
- [8] C. Lester, D. Summers, Measuring masses of semiinvisibly decaying particles pair produced at hadron colliders, Phys. Lett. B463 (1999) 99–103. [arXiv:hep-ph/9906349](#).
- [9] A. Barr, C. Lester, P. Stephens, m_{T2} : The Truth behind the glamour, J. Phys. G29 (2003) 2343–2363. [arXiv:hep-ph/0304226](#).
- [10] H.-C. Cheng, Z. Han, Minimal Kinematic Constraints and m_{T2} , JHEP 0812 (2008) 063. [arXiv:0810.5178](#).
- [11] Y. Bai, H.-C. Cheng, J. Gallicchio, J. Gu, Stop the Top Background of the Stop Search, JHEP 1207 (2012) 110. [arXiv:1203.4813](#).
- [12] D. R. Tovey, On measuring the masses of pair-produced semi-invisibly decaying particles at hadron colliders, JHEP 0804 (2008) 034. [arXiv:0802.2879](#).
- [13] G. Polesello, D. R. Tovey, Supersymmetric particle mass measurement with the boost-corrected contranverse mass, JHEP 1003 (2010) 030. [arXiv:0910.0174](#).

- [14] K. T. Matchev, M. Park, A General method for determining the masses of semi-invisibly decaying particles at hadron colliders, *Phys. Rev. Lett.* 107 (2011) 061801. [arXiv:0910.1584](#).
- [15] M. L. Graesser, J. Shelton, Hunting Mixed Top Squark Decays, *Phys. Rev. Lett.* 111 (12) (2013) 121802. [arXiv:1212.4495](#).
- [16] M. R. Buckley, J. D. Lykken, C. Rogan, M. Spiropulu, Super-Razor and Searches for Stopped and Charginos at the LHC, *Phys. Rev. D* 89 (5) (2014) 055020. [arXiv:1310.4827](#).
- [17] G. Aad, et al., Combined Measurement of the Higgs Boson Mass in pp Collisions at $\sqrt{s} = 7$ and 8 TeV with the ATLAS and CMS Experiments, *Phys. Rev. Lett.* 114 (2015) 191803. [arXiv:1503.07589](#).
- [18] G. Aad, et al., Measurements of the Higgs boson production and decay rates and constraints on its couplings from a combined ATLAS and CMS analysis of the LHC pp collision data at $\sqrt{s} = 7$ and 8 TeV, *JHEP* 08 (2016) 045. [arXiv:1606.02266](#).
- [19] D. Alves, Simplified Models for LHC New Physics Searches, *J. Phys. G* 39 (2012) 105005. [arXiv:1105.2838](#).
- [20] S. Kraml, S. Kulkarni, U. Laa, A. Lessa, W. Magerl, D. Proschofsky-Spindler, W. Waltenberger, SModelS: a tool for interpreting simplified-model results from the LHC and its application to supersymmetry, *Eur. Phys. J. C* 74 (2014) 2868. [arXiv:1312.4175](#).
- [21] S. Kraml, S. Kulkarni, U. Laa, A. Lessa, V. Magerl, W. Magerl, D. Proschofsky-Spindler, M. Traub, W. Waltenberger, SModelS v1.0: a short user guide. [arXiv:1412.1745](#).
- [22] M. Papucci, K. Sakurai, A. Weiler, L. Zeune, Fastlim: a fast LHC limit calculator, *Eur. Phys. J. C* 74 (11) (2014) 3163. [arXiv:1402.0492](#).
- [23] D. Barducci, A. Belyaev, M. Buchkremer, J. Marrouche, S. Moretti, L. Panizzi, XQCAT: eXtra Quark Combined Analysis Tool, *Comput. Phys. Commun.* 197 (2015) 263–275. [arXiv:1409.3116](#).
- [24] S. Caron, J. S. Kim, K. Rolbiecki, R. Ruiz de Austri, B. Stienen, The BSM-AI project: SUSY-AI - Generalizing LHC limits on Supersymmetry with Machine Learning. [arXiv:1605.02797](#).
- [25] K. Cranmer, I. Yavin, RECAST: Extending the Impact of Existing Analyses, *JHEP* 04 (2011) 038. [arXiv:1010.2506](#).
- [26] M. Drees, H. Dreiner, D. Schmeier, J. Tattersall, J. S. Kim, CheckMATE: Confronting your Favourite New Physics Model with LHC Data, *Comput. Phys. Commun.* 187 (2015) 227–265. [arXiv:1312.2591](#).
- [27] J. S. Kim, D. Schmeier, J. Tattersall, K. Rolbiecki, A framework to create customised LHC analyses within CheckMATE, *Comput. Phys. Commun.* 196 (2015) 535–562. [arXiv:1503.01123](#).
- [28] E. Conte, B. Fuks, G. Serret, MadAnalysis 5, A User-Friendly Framework for Collider Phenomenology, *Comput. Phys. Commun.* 184 (2013) 222–256. [arXiv:1206.1599](#).
- [29] B. Dumont, B. Fuks, S. Kraml, S. Bein, G. Chalons, E. Conte, S. Kulkarni, D. Sengupta, C. Wymant, Toward a public analysis database for LHC new physics searches using MADANALYSIS 5, *Eur. Phys. J. C* 75 (2) (2015) 56. [arXiv:1407.3278](#).
- [30] J. M. Butterworth, D. Grellscheid, M. Krämer, D. Yallup, Constraining new physics with collider measurements of Standard Model signatures. [arXiv:1606.05296](#).
- [31] A. Buckley, J. Butterworth, L. Lonnblad, D. Grellscheid, H. Hoeth, J. Monk, H. Schulz, F. Siegert, Rivet user manual, *Comput. Phys. Commun.* 184 (2013) 2803–2819. [arXiv:1003.0694](#).
- [32] P. Z. Skands, et al., SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators, *JHEP* 07 (2004) 036. [arXiv:hep-ph/0311123](#).
- [33] B. C. Allanach, et al., SUSY Les Houches Accord 2, *Comput. Phys. Commun.* 180 (2009) 8–25. [arXiv:0801.0045](#).
- [34] N. D. Christensen, C. Duhr, FeynRules - Feynman rules made easy, *Comput. Phys. Commun.* 180 (2009) 1614–1641. [arXiv:0806.4194](#).
- [35] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, B. Fuks, FeynRules 2.0 - A complete toolbox for tree-level phenomenology, *Comput. Phys. Commun.* 185 (2014) 2250–2300. [arXiv:1310.1921](#).
- [36] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter, UFO - The Universal FeynRules Output, *Comput. Phys. Commun.* 183 (2012) 1201–1214. [arXiv:1108.2040](#).
- [37] Wikipedia, Fritz (chess), [Online; accessed May 20, 2016].
URL [https://en.wikipedia.org/wiki/Fritz_\(chess\)](https://en.wikipedia.org/wiki/Fritz_(chess))
- [38] C. GmbH, ChessBase, [Online; accessed May 20, 2016].
URL <http://en.chessbase.com/>
- [39] W. Beenakker, R. Hopker, M. Spira, P. M. Zerwas, Squark and gluino production at hadron colliders, *Nucl. Phys. B* 492 (1997) 51–103. [arXiv:hep-ph/9610490](#).
- [40] W. Beenakker, M. Krämer, T. Plehn, M. Spira, P. M. Zerwas, Stop production at hadron colliders, *Nucl. Phys. B* 515 (1998) 3–14. [arXiv:hep-ph/9710451](#).
- [41] W. Beenakker, M. Klasen, M. Krämer, T. Plehn, M. Spira, P. M. Zerwas, The Production of charginos/neutralinos and sleptons at hadron colliders, *Phys. Rev. Lett.* 83 (1999) 3780–3783, [Erratum: *Phys. Rev. Lett.* 100, 029901 (2008)]. [arXiv:hep-ph/9906298](#).
- [42] M. Spira, Higgs and SUSY particle production at hadron colliders, in: *Supersymmetry and unification of fundamental interactions. Proceedings, 10th International Conference, SUSY'02, Hamburg, Germany, June 17–23, 2002*, pp. 217–226. [arXiv:hep-ph/0211145](#).
- [43] T. Plehn, Measuring the MSSM Lagrangean, *Czech. J. Phys.* 55 (2005) B213–B220. [arXiv:hep-ph/0410063](#).
- [44] A. Kulesza, L. Motyka, Threshold resummation for squark-antisquark and gluino-pair production at the LHC, *Phys. Rev. Lett.* 102 (2009) 111802. [arXiv:0807.2405](#).
- [45] A. Kulesza, L. Motyka, Soft gluon resummation for the production of gluino-gluino and squark-antisquark pairs at the LHC, *Phys. Rev. D* 80 (2009) 095004. [arXiv:0905.4749](#).
- [46] W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, et al., Soft-gluon resummation for squark and gluino

- hadroproduction, JHEP 0912 (2009) 041. [arXiv:0909.4418](#).
- [47] W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, et al., Supersymmetric top and bottom squark production at hadron colliders, JHEP 1008 (2010) 098. [arXiv:1006.4771](#).
 - [48] W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, et al., Squark and Gluino Hadroproduction, Int. J. Mod. Phys. A26 (2011) 2637–2664. [arXiv:1105.1110](#).
 - [49] T. Sjöstrand, S. Mrenna, P. Z. Skands, A Brief Introduction to PYTHIA 8.1, Comput. Phys. Commun. 178 (2008) 852–867. [arXiv:0710.3820](#).
 - [50] The Pythia 8 Collaboration, PYTHIA 8 Online Documentation, [Online; accessed May 12, 2016]. URL <http://home.thep.lu.se/~torbjorn/pythia82html/Welcome.html>
 - [51] W. Kilian, T. Ohl, J. Reuter, WHIZARD: Simulating Multi-Particle Processes at LHC and ILC, Eur. Phys. J. C71 (2011) 1742. [arXiv:0708.4233](#).
 - [52] M. Moretti, T. Ohl, J. Reuter, O'Mega: An Optimizing matrix element generator. [arXiv:hep-ph/0102195](#).
 - [53] A. Belyaev, N. D. Christensen, A. Pukhov, CalcHEP 3.4 for collider physics within and beyond the Standard Model, Comput. Phys. Commun. 184 (2013) 1729–1769. [arXiv:1207.6082](#).
 - [54] Mini-workshop on recasting ATLAS and CMS new physics searches. URL <https://lpsc-indico.in2p3.fr/Indico/event/1085/>
 - [55] K. A. Olive, et al., Review of Particle Physics, Chin. Phys. C38 (2014) 090001.
 - [56] J. S. Kim, K. Rolbiecki, K. Sakurai, J. Tattersall, 'Stop' that ambulance! New physics at the LHC?, JHEP 12 (2014) 010. [arXiv:1406.0858](#).
 - [57] R. D. Ball, et al., Parton distributions with LHC data, Nucl. Phys. B867 (2013) 244–289. [arXiv:1207.1303](#).
 - [58] G. Aad, et al., Search for squarks and gluinos with the ATLAS detector in final states with jets and missing transverse momentum using $\sqrt{s} = 8$ TeV proton–proton collision data, JHEP 09 (2014) 176. [arXiv:1405.7875](#).
 - [59] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, T. Stelzer, MadGraph 5: Going Beyond, JHEP 1106 (2011) 128. [arXiv:1106.0522](#).
 - [60] G. Aad, et al., Search for new phenomena in final states with an energetic jet and large missing transverse momentum in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, Eur. Phys. J. C75 (7) (2015) 299, [Erratum: Eur. Phys. J. C75, no. 9, 408 (2015)]. [arXiv:1502.01518](#).
 - [61] S. Chatrchyan, et al., Search for supersymmetry in hadronic final states with missing transverse energy using the variables α_T and b-quark multiplicity in pp collisions at $\sqrt{s} = 8$ TeV, Eur. Phys. J. C73 (9) (2013) 2568. [arXiv:1303.2985](#).
 - [62] G. Aad, et al., Search for new phenomena in final states with large jet multiplicities and missing transverse momentum at $\sqrt{s} = 8$ TeV proton-proton collisions using the ATLAS experiment, JHEP 10 (2013) 130, [Erratum: JHEP 01 (2014) 109]. [arXiv:1308.1841](#).
 - [63] J. Cao, L. Shang, J. M. Yang, Y. Zhang, Explanation of the ATLAS Z-Peaked Excess in the NMSSM, JHEP 06 (2015) 152. [arXiv:1504.07869](#).
 - [64] G. Aad, et al., Search for direct third-generation squark pair production in final states with missing transverse momentum and two b-jets in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, JHEP 1310 (2013) 189. [arXiv:1308.2631](#).
 - [65] G. Aad, et al., Search for direct production of charginos and neutralinos in events with three leptons and missing transverse momentum in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, JHEP 1404 (2014) 169. [arXiv:1402.7029](#).
 - [66] G. Aad, et al., Search for direct top-squark pair production in final states with two leptons in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, JHEP 06 (2014) 124. [arXiv:1403.4853](#).
 - [67] G. Aad, et al., Search for direct top squark pair production in events with a Z boson, b-jets and missing transverse momentum in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, Eur. Phys. J. C74 (6) (2014) 2883. [arXiv:1403.5222](#).
 - [68] G. Aad, et al., Search for supersymmetry at $\sqrt{s} = 8$ TeV in final states with jets and two same-sign leptons or three leptons with the ATLAS detector, JHEP 06 (2014) 035. [arXiv:1404.2500](#).
 - [69] G. Aad, et al., Search for top squark pair production in final states with one isolated lepton, jets, and missing transverse momentum in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, JHEP 1411 (2014) 118. [arXiv:1407.0583](#).
 - [70] G. Aad, et al., Search for pair-produced third-generation squarks decaying via charm quarks or in compressed supersymmetric scenarios in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, Phys. Rev. D90 (5) (2014) 052008. [arXiv:1407.0608](#).
 - [71] G. Aad, et al., Search for new phenomena in events with a photon and missing transverse momentum in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, Phys. Rev. D91 (1) (2015) 012008, [Erratum: Phys. Rev. D92, no. 5, 059903 (2015)]. [arXiv:1411.1559](#).
 - [72] G. Aad, et al., Search for direct pair production of a chargino and a neutralino decaying to the 125 GeV Higgs boson in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, Eur. Phys. J. C75 (5) (2015) 208. [arXiv:1501.07110](#).
 - [73] G. Aad, et al., Search for supersymmetry in events containing a same-flavour opposite-sign dilepton pair, jets, and large missing transverse momentum in $\sqrt{s} = 8$ TeV pp collisions with the ATLAS detector, Eur. Phys. J. C75 (7) (2015) 318. [arXiv:1503.03290](#).
 - [74] G. Aad, et al., ATLAS Run 1 searches for direct pair production of third-generation squarks at the Large Hadron Collider, Eur. Phys. J. C75 (10) (2015) 510, [Erratum: Eur. Phys. J. C76, no. 3, 153 (2016)]. [arXiv:1506.08616](#).
 - [75] Search for supersymmetry at $\sqrt{s} = 8$ TeV in final states with jets, missing transverse momentum and one isolated lepton, Tech. Rep. ATLAS-CONF-2012-104, CERN, Geneva (Aug 2012).
 - [76] Search for New Phenomena in Monojet plus Missing Transverse Momentum Final States using 10 fb⁻¹ of pp Collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector at the LHC, Tech. Rep. ATLAS-CONF-2012-147, CERN, Geneva (Nov 2012).
 - [77] Search for direct production of the top squark in the all-hadronic $t\bar{t} + E_T^{\text{miss}}$ final state in 21 fb⁻¹ of pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, Tech. Rep. ATLAS-CONF-2013-024, CERN, Geneva (Mar 2013).

- [78] Search for direct-slepton and direct-chargino production in final states with two opposite-sign leptons, missing transverse momentum and no jets in 20/fb of pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector, Tech. Rep. ATLAS-CONF-2013-049, CERN, Geneva (May 2013).
- [79] Search for strong production of supersymmetric particles in final states with missing transverse momentum and at least three b -jets using 20.1 fb $^{-1}$ of pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS Detector, Tech. Rep. ATLAS-CONF-2013-061, CERN, Geneva (Jun 2013).
- [80] The ATLAS Collaboration, Search for strongly produced supersymmetric particles in decays with two leptons at $\sqrt{s} = 8$ TeV, Tech. Rep. ATLAS-CONF-2013-089, CERN, Geneva (Aug 2013).
- [81] The ATLAS Collaboration, Search for an Invisibly Decaying Higgs Boson Produced via Vector Boson Fusion in pp Collisions at $\sqrt{s} = 8$ TeV using the ATLAS Detector at the LHC, Tech. Rep. ATLAS-CONF-2015-004, CERN, Geneva (Mar 2015).
- [82] V. Khachatryan, et al., Search for dark matter, extra dimensions, and unparticles in monojet events in proton–proton collisions at $\sqrt{s} = 8$ TeV, Eur. Phys. J. C75 (5) (2015) 235. [arXiv:1408.3583](#).
- [83] V. Khachatryan, et al., Search for Physics Beyond the Standard Model in Events with Two Leptons, Jets, and Missing Transverse Momentum in pp Collisions at $\sqrt{s} = 8$ TeV, JHEP 04 (2015) 124. [arXiv:1502.06031](#).
- [84] V. Khachatryan, et al., Search for the production of dark matter in association with top-quark pairs in the single-lepton final state in proton-proton collisions at $\sqrt{s} = 8$ TeV, JHEP 06 (2015) 121. [arXiv:1504.03198](#).
- [85] S. Baek, P. Ko, P. Wu, Top-philic Scalar Dark Matter with a Vector-like Fermionic Top Partner, JHEP 10 (2016) 117. [arXiv:1606.00072](#).
- [86] The CMS Collaboration, Search for new physics in events with same-sign dileptons and jets in pp collisions at 8 TeV, Tech. Rep. CMS-PAS-SUS-13-013 (2013).
- [87] G. Aad, et al., Search for supersymmetry at $\sqrt{s} = 13$ TeV in final states with jets and two same-sign leptons or three leptons with the ATLAS detector, Eur. Phys. J. C76 (5) (2016) 259. [arXiv:1602.09058](#).
- [88] M. Aaboud, et al., Search for new phenomena in events with a photon and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector, JHEP 06 (2016) 059. [arXiv:1604.01306](#).
- [89] M. Aaboud, et al., Search for new phenomena in final states with an energetic jet and large missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector, Phys. Rev. D94 (3) (2016) 032005. [arXiv:1604.07773](#).
- [90] M. Aaboud, et al., Search for squarks and gluinos in final states with jets and missing transverse momentum at $\sqrt{s} = 13$ TeV with the ATLAS detector, Eur. Phys. J. C76 (7) (2016) 392. [arXiv:1605.03814](#).
- [91] G. Aad, et al., Search for gluinos in events with an isolated lepton, jets and missing transverse momentum at $\sqrt{s} = 13$ TeV with the ATLAS detector, Eur. Phys. J. C76 (10) (2016) 565. [arXiv:1605.04285](#).
- [92] G. Aad, et al., Search for pair production of gluinos decaying via stop and sbottom in events with b -jets and large missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector, Phys. Rev. D94 (3) (2016) 032003. [arXiv:1605.09318](#).
- [93] M. Aaboud, et al., Search for top squarks in final states with one isolated lepton, jets, and missing transverse momentum in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector, Phys. Rev. D94 (5) (2016) 052009. [arXiv:1606.03903](#).
- [94] A search for Supersymmetry in events containing a leptonically decaying Z boson, jets and missing transverse momentum in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector, Tech. Rep. ATLAS-CONF-2015-082, CERN, Geneva (Dec 2015). URL <http://cds.cern.ch/record/2114854>
- [95] Search for production of vector-like top quark pairs and of four top quarks in the lepton-plus-jets final state in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector, Tech. Rep. ATLAS-CONF-2016-013, CERN, Geneva (Mar 2016). URL <http://cds.cern.ch/record/2140998>
- [96] Search for top squarks in final states with one isolated lepton, jets, and missing transverse momentum in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector, Tech. Rep. ATLAS-CONF-2016-050, CERN, Geneva (Aug 2016). URL <https://cds.cern.ch/record/2206132>
- [97] Search for direct top squark pair production and dark matter production in final states with two leptons in $\sqrt{s} = 13$ TeV pp collisions using 13.3 fb $^{-1}$ of ATLAS data, Tech. Rep. ATLAS-CONF-2016-076, CERN, Geneva (Aug 2016). URL <http://cds.cern.ch/record/2206249>
- [98] Search for new physics in final states with two opposite-sign same-flavor leptons, jets and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV, Tech. Rep. CMS-PAS-SUS-15-011, CERN, Geneva (2015). URL <https://cds.cern.ch/record/2114811>
- [99] Search for Supersymmetry at the high luminosity LHC with the ATLAS experiment, Tech. Rep. ATL-PHYS-PUB-2014-010, CERN, Geneva (Jul 2014). URL <http://cds.cern.ch/record/1735031>
- [100] Prospects for benchmark Supersymmetry searches at the high luminosity LHC with the ATLAS Detector, Tech. Rep. ATL-PHYS-PUB-2013-011, CERN, Geneva (Sep 2013). URL <http://cds.cern.ch/record/1604505>
- [101] A. J. Barr, C. G. Lester, A Review of the Mass Measurement Techniques proposed for the Large Hadron Collider, J. Phys. G37 (2010) 123001. [arXiv:1004.2732](#).
- [102] W. van Neerven, J. Vermaseren, K. Gaemers, Lepton - jet events as a signature for W production in p anti- p collisions, Tech. rep. (1982).
- [103] G. Arnison, et al., Experimental Observation of Isolated Large Transverse Energy Electrons with Associated Missing Energy at $\sqrt{s} = 540$ GeV, Phys. Lett. B122 (1983) 103–116.
- [104] M. Banner, et al., Observation of Single Isolated Electrons of High Transverse Momentum in Events with Missing Transverse Energy at the CERN anti- p p Collider, Phys. Lett. B122 (1983) 476–485.

- [105] J. Smith, W. van Neerven, J. Vermaseren, The Transverse Mass and Width of the W Boson, Phys. Rev. Lett. 50 (1983) 1738.
- [106] V. D. Barger, T. Han, R. Phillips, Improved Transverse Mass Variable for Detecting Higgs Boson Decays Into Z Pairs, Phys. Rev. D36 (1987) 295.
- [107] L. Randall, D. Tucker-Smith, Dijet Searches for Supersymmetry at the LHC, Phys. Rev. Lett. 101 (2008) 221803. [arXiv:0806.1049](#).
- [108] V. Khachatryan, et al., Search for Supersymmetry in pp Collisions at 7 TeV in Events with Jets and Missing Transverse Energy, Phys. Lett. B698 (2011) 196–218. [arXiv:1101.1628](#).
- [109] C. Rogan, Kinematical variables towards new dynamics at the LHC [arXiv:1006.2727](#).
- [110] S. Chatrchyan, et al., Inclusive search for squarks and gluinos in pp collisions at $\sqrt{s} = 7$ TeV, Phys. Rev. D85 (2012) 012004. [arXiv:1107.1279](#).
- [111] V. Khachatryan, et al., Search for Supersymmetry Using Razor Variables in Events with b -Tagged Jets in pp Collisions at $\sqrt{s} = 8$ TeV, Phys. Rev. D91 (2015) 052018. [arXiv:1502.00300](#).
- [112] C. Chen, New approach to identifying boosted hadronically-decaying particle using jet substructure in its center-of-mass frame, Phys. Rev. D85 (2012) 034007. [arXiv:1112.2567](#).
- [113] R. Brun, F. Rademakers, ROOT: An object oriented data analysis framework, Nucl. Instrum. Meth. A389 (1997) 81–86.
- [114] M. Dobbs, J. B. Hansen, The HepMC C++ Monte Carlo event record for High Energy Physics, Comput. Phys. Commun. 134 (2001) 41–46.
- [115] K. Cranmer, Practical Statistics for the LHC, in: Proceedings, 2011 European School of High-Energy Physics (ESHEP 2011), 2015, pp. 267–308, [247(2015)]. [arXiv:1503.07622](#).
URL <https://inspirehep.net/record/1356277/files/arXiv:1503.07622.pdf>
- [116] S. S. Wilks, The large-sample distribution of the likelihood ratio for testing composite hypotheses, Ann. Math. Statist. 9 (1) (1938) 60–62.
URL <http://dx.doi.org/10.1214/aoms/1177732360>
- [117] M. Dowell, P. Jarratt, The “Pegasus” method for computing the root of an equation, BIT Numerical Mathematics 12 (4) (1972) 503–508.
URL <http://dx.doi.org/10.1007/BF01932959>
- [118] Electron efficiency measurements with the ATLAS detector using the 2015 LHC proton-proton collision data, Tech. Rep. ATLAS-CONF-2016-024, CERN, Geneva (Jun 2016).
URL <http://cds.cern.ch/record/2157687>
- [119] Electron identification efficiency measured with $Z \rightarrow ee$ events using 2016 data, [Online; accessed 12-Nov-2016].
URL <https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/EGAM-2016-002/>
- [120] Performance assumptions for an upgraded ATLAS detector at a High-Luminosity LHC, Tech. Rep. ATL-PHYS-PUB-2013-004, CERN, Geneva (Mar 2013).
URL <https://cds.cern.ch/record/1527529>
- [121] G. Aad, et al., Muon reconstruction performance of the ATLAS detector in proton-proton collision data at $\sqrt{s} = 13$ TeV, Eur. Phys. J. C76 (5) (2016) 292. [arXiv:1603.05598](#).
- [122] Performance assumptions based on full simulation for an upgraded ATLAS detector at a High-Luminosity LHC, Tech. Rep. ATL-PHYS-PUB-2013-009, CERN, Geneva (Sep 2013).
URL <http://cds.cern.ch/record/1604420>
- [123] Measurement of the b -tag Efficiency in a Sample of Jets Containing Muons with 5 fb^{-1} of Data from the ATLAS Detector, Tech. Rep. ATLAS-CONF-2012-043, CERN, Geneva (Mar 2012).
- [124] Measuring the b -tag efficiency in a top-pair sample with 4.7 fb^{-1} of data from the ATLAS detector, Tech. Rep. ATLAS-CONF-2012-097, CERN, Geneva (Jul 2012).
- [125] Measurement of the Mistag Rate with 5 fb^{-1} of Data Collected by the ATLAS Detector, Tech. Rep. ATLAS-CONF-2012-040, CERN, Geneva (Mar 2012).
- [126] Expected performance of the ATLAS b -tagging algorithms in Run-2, Tech. Rep. ATL-PHYS-PUB-2015-022, CERN, Geneva (Jul 2015).
URL <http://cds.cern.ch/record/2037697>